Annual Report

FY 1975
Vol. I

Speech Evaluation

30 June 1975

Prepared for the Defense Communications Agency
under Electronic Systems Division Contract F19628-73-C-0002 by

Lincoln Laboratory

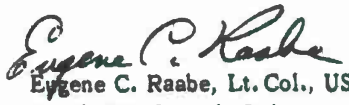
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LEXINGTON, MASSACHUSETTS

The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology, for the Military Satellite Office of the Defense Communications Agency under Air Force Contract F19628-73-C-0002.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER



Eugene C. Raabe, Lt. Col., USAF
Chief, ESD Lincoln Laboratory Project Office

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

SPEECH EVALUATION

ANNUAL REPORT FOR FY 1975
TO THE
DEFENSE COMMUNICATIONS AGENCY

VOLUME I

30 JUNE 1975

ISSUED 14 JANUARY 1976

LEXINGTON MASSACHUSETTS

ABSTRACT

Volume I reports the work performed during FY 75 on the DCA-Speech Evaluation Contract. Two broad categories of work are described: algorithmic studies and hardware design.

Several algorithms for digitizing and reducing the data rate of speech signals are described. These algorithms include an adaptive residual coder (ARC) designed to produce data at 16 and 9.6 kbps, an adaptive predictive coder (APC) at 8 kbps, a voice-excited linear predictor (VELP) at 8 kbps, and a straight linear predictive coded (LPC) vocoder at 2.4, 3.6, and 4.8 kbps. In addition, some work on pitch or excitation extraction is described. All these studies are evaluated on a real-time facility which is described.

In the hardware design area, the digital voice terminal is described in detail, as well as some follow-on next-generation LSI studies.

Volume II will contain a DVT manual, program listings, and a cross assembler.

CONTENTS

GLOSSARY

AC	Accumulator
ADPCM	Adaptive Differential Pulse Code Modulation
ALU	Arithmetic/Logic Unit
AMDF	Absolute Magnitude Difference Function
APC	Adaptive Predictive Coding
ARC	Adaptive Residual Coding
CPE	Central Processing Element
DCA	Defense Communications Agency
DFT	Discrete Fourier Transform
DIP	Dual In-Line Package
DVT	Digital Voice Terminal
ECL	Emitter Coupled Logic
EFL	Emitter Follower Logic
FDP	Fast Digital Processor
IC	Integrated Circuit
LDVT	Lincoln Digital Voice Terminal
LPC	Linear Predictive Coding
LSI	Large-Scale Integration
MSI	Medium-Scale Integration
PCM	Pulse Code Modulation
RAM	Random Access Memory
ROM	Read-Only Memory
SSI	Small-Scale Integration
TTL	Transistor-Transistor Logic
VLSI	Very Large-Scale Integration

ACKNOWLEDGMENTS

This report for FY 1975 to the Defense Communications Agency was edited by J. Tierney. It is based on work by P.E. Blankenship, B. Gold, E.M. Hofstetter, M. L. Malpass, S. Seneff, and J. Tierney. Thanks are also due to R. Sonderegger of Defense Communications Engineering Center for continuing technical interaction.

SPEECH EVALUATION

I. PROGRAM OVERVIEW – FY 1975

The goal of Lincoln's work for the Defense Communications Agency in FY 75 was the development and evaluation of various speech digitization algorithms using the real-time speech-processing facility centered about the fast digital processor (FDP). As the program unfolded, it became clear that an additional focus was the transplanting of these algorithms onto the digital voice terminals (DVTs) and their subsequent testing by the Narrow-Band Speech Consortium's Test and Evaluation program. Major aspects of this program included:

(a) An attempt to improve the adaptive predictive coding (APC) algorithm previously developed by G.T.&E., Sylvania.

(b) A real-time simulation of the adaptive residual coding (ARC) hardware constructed by Codex.

(c) Development of Lincoln's version of the autocorrelation method for linear predictive coding (LPC).

The eventual results of these tasks can be summed up best as follows: LPC at 2400, 3600, and 4800 bps, APC at 8000 bps, and ARC at 9600 and 16,000 bps were implemented on the DVTs for use by the Consortium's test program. A significant finding was the demonstration that the DVT was a versatile speech processor that could be quickly and easily programmed to implement a variety of speech algorithms. Since the DVT, constructed with present technology, is a quite compact device containing about 500 commercial integrated circuit packages, this finding indicates that speech digitizers in the near future can be both small and cheap, yet flexible.

At the suggestion of R. Sonderegger, the major results of our efforts were presented at the EASCON Conference on 30 September 1975 in Washington, D.C., and published in the EASCON Proceedings. The publications include a description of the DVT, a description of the pitch detector used for the LPC algorithms, and descriptions of the APC, LPC, and ARC algorithms.

The 1970's appear to herald a greater degree of activity in speech terminal development. The technical reasons for this stem from the development of new speech-processing algorithms such as LPC and APC plus the rapid advance of technology which promises to lead to improved and cheaper speech terminals. A new and potentially important direction has to do with the establishment of communication networks with both speech and data transmission capabilities. The eventual form that such networks take is still open but it is already clear that digital communications for both voice and data could benefit greatly from the development of reliable and good quality speech terminals running at rates appreciably less than the 64-kilobit channels designated by the Bell System. Thus, we expect that further development of speech algorithms and their efficient implementation will be worth pursuing for at least 5 more years.

In the 1960's, the only practical narrow-band speech device was the channel vocoder. While this device is still a respectable speech digitizer, there is reason to believe that the LPC algorithm is an improvement both in ease of implementation and also in improved speech quality, despite the fact that LPC has been developed only within the past 5 years whereas channel vocoders have a 40-year history of development.

II. OUTLINE OF SPECIFIC TASKS

Volume I of Lincoln Laboratory's FY 75 work on speech evaluation devotes a separate section to each of the major areas of effort. Section III presents a general description of the fast digital processor (FDP) facility for real-time simulation of speech-compression algorithms. This facility was the vehicle for algorithmic research on ARC, APC, LPC, and pitch extraction. In Sec. IV, we present the design motivation for the digital voice terminal (DVT) – a small, fast, versatile signal-processing computer capable of real-time performance when running DCA speech-compression algorithms. The DVT approach to machine design was distinctly different than that used for the FDP, and the structure is very much simpler. Section V then presents some research along the direction of less-ambitious lower-power machines that can run specific speech-compression algorithms (e.g., LPC at a 2400-bps rate) and lend themselves to large-scale integration (LSI) implementation. After dealing with the issues of machine design in Secs. III through V, the report moves to the topic of speech-compression algorithms (voice coders). Sections VI through VIII discuss the three algorithms (ARC, APC, LPC) studied on the FDP and finally implemented on the DVT. All three of these algorithms were tested at various data rates under the narrow-band consortium test and evaluation effort.

Section IX discusses the problem of pitch detection and implementing the time-domain Gold pitch detector on the FDP and DVT. In addition, a comparison between the time-domain detector and an absolute magnitude difference function detector is made in terms of program length and running time. No statements are made about the quality of synthesized vocoder speech using either pitch detector, as these issues are better left to listening tests.

Finally, Section X presents an overview of the program and a brief summary of our ongoing efforts.

Volume II of this annual report consists of a DVT manual, Fortran cross assembler listing, diagnostic listings, and annotated listings for LPC, ARC, and APC algorithms.

III. THE FAST DIGITAL PROCESSOR (FDP) SIMULATION FACILITY

The simulation in real time of complicated algorithms for speech bandwidth compression is a relatively new approach to speech-compression research. The FDP was designed and constructed at Lincoln Laboratory to run such simulations, and has done so quite successfully since 1970. In fact, it may well have been the first general-purpose stored-program computer to run a complicated speech program in real time so that people could actually talk "through it" and evaluate the particular algorithm being run. It was the FDP facility that was used as the tool for the Lincoln Laboratory speech evaluation DCA effort in early FY 75. By using this facility, it was possible to code up various forms of APC, ARC, and LPC algorithms, as well as two disparate pitch detectors. These could be run as real-time code on the FDP, and, along with an input A/D converter and an output D/A converter, allow the machine to appear as a "black box" speech coder configured as needed for people to use as shown in Fig. III-1. With careful listening tests, it was possible to try many parameter variations and optimize each of the algorithms for certain conditions of data rate and bandwidth. The following is a discussion of the facility at Lincoln Laboratory, indicating the peripheral devices particularly important for speech-compression algorithmic optimization.

Figure III-2 is a general block diagram of the facility. The core element in the facility is of course the FDP,* a signal-processing emitter coupled-logic computer with a cycle time of 150 nsec, 18-bit data word, separate program memory (5k) and data memory (4k × 2), four parallel arithmetic units each containing a full array multiplier, and a double-width (36 bits) program word. The machine was designed to perform a complex multiply or digital filter recursion (each requiring four multiplies) in the order of 1 μsec, including setup time for the four arithmetic elements. The 36-bit program word width allows control of the four individual arithmetic elements as well as memory read and write or control simultaneously. The data memory is actually two separate 4k memories in order to allow for complex data manipulation. As a result of the parallelism, separate data and program memory, and instruction overlap at the 150-nsec rate, the FDP was between a factor of 10 and 100 faster than other contemporary machines. This fact enabled speech research to be conducted using real-time simulations.

The FDP program and data memory are loaded through an I-O connection to a Univac 1219 machine with 32k of 18-bit word store, a 2-μsec cycle time, and eight I-O channels of its own. The Univac is connected to a 230k word drum, paper tape reader-punch, typewriter, two display scopes, two Ampex 7-track digital tape drives, and A/D and D/A converters. The FDP is in effect a peripheral device to the 1219, but once code is loaded into FDP memory, the FDP runs as a freestanding machine with its own set of A/D and D/A converters for input and output of analog speech.

The Univac 1219 is the background processor for assembly, editing, debugging, display of data blocks, and running of simple tasks that are logically performed outside of the FDP.

A more complex real-time arrangement allows for several speech algorithms to reside in binary form on the Univac magnetic drum, which can be read into Univac core and then over to FDP memory in a few tens of milliseconds. In this fashion, it is possible to be talking through the FDP as shown in Fig. III-1 with one algorithm in use, and select by way of the keyboard a

*B. Gold, I. L. Lebow, P. G. McHugh, and C. M. Rader, "The FDP, A Fast Programmable Signal Processor," IEEE Trans. Computers C-20, 33 (1971), DDC AD-728092.

second algorithm or parameter variation and have it loaded and running while two users notice only a minor "click." This is, in fact, the mode used to refine our algorithms for APC, ARC, and LPC before transferring these codes to DVT language.

The FDP facility continues to be used for speech research, although the new DVT devices are in several ways simpler to program, though considerably less flexible in terms of debugging features.

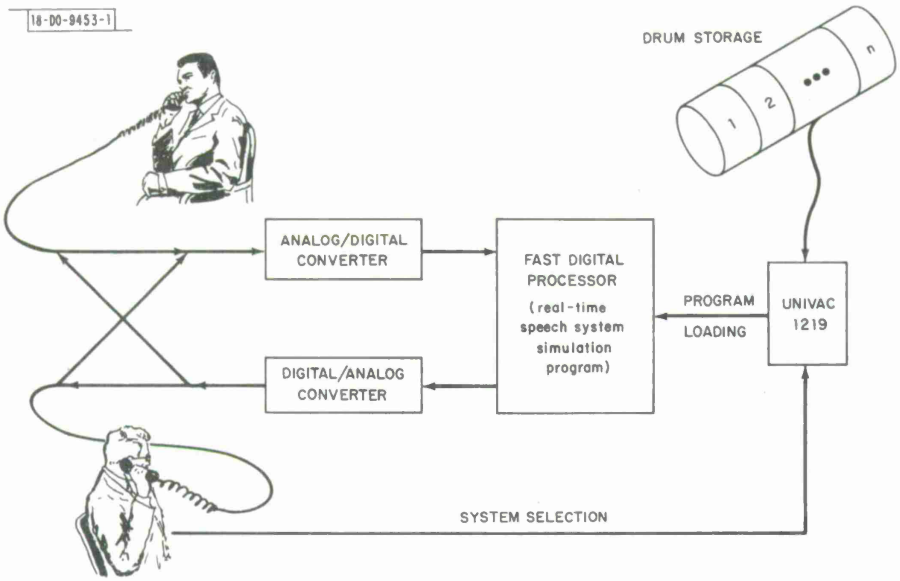


Fig. III-1. Real-time conversational evaluation of simulated speech-compression systems.

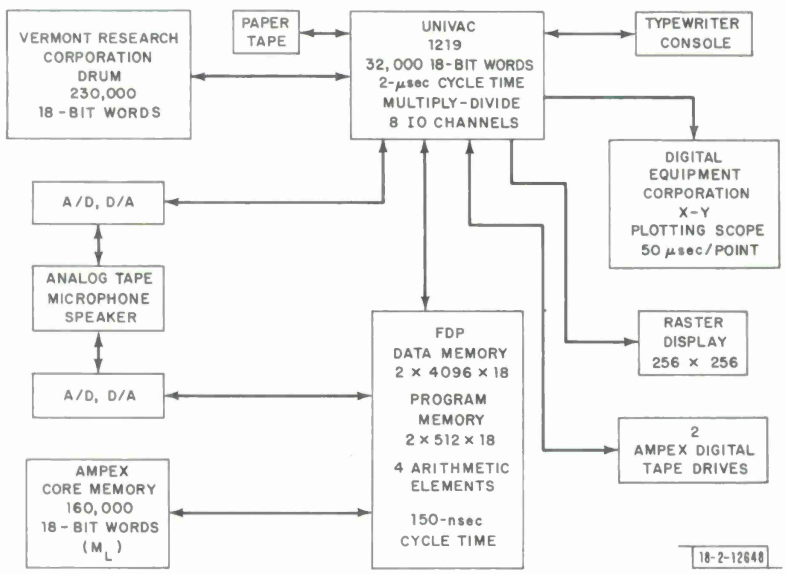


Fig. III-2. FDP-1219 speech-processing facility.

IV. DIGITAL VOICE TERMINAL (DVT) DESIGN

A. INTRODUCTION

Inherent with the large, complex, expensive, one-of-a-kind FDP facility are programming difficulties and the inability to operate in a stand-alone mode. These shortcomings indicated that a second-generation processor was needed. A compact, easy-to-use, easy-to-replicate, relatively inexpensive facility capable of stand-alone operation and of equivalent or superior performance capability to FDP became the overall objective.

The Lincoln Digital Voice Terminal (LDVT) was designed to meet this objective. Comprised of a custom-designed 55-nsec, 16-bit minicomputer and appropriate integrated peripherals, the LDVT has proven to be well matched to the real-time speech-processing problem. In this section, a technical description of the LDVT system is presented. The genuine power and versatility of the processor are illustrated in later sections of this report via detailed descriptions of three fully operational vocoder software packages that have been written for it. These algorithms include linear predictive coding (LPC), adaptive predictive coding (APC), and adaptive residual coding (ARC).

B. LDVT SYSTEM DESCRIPTION

1. Minicomputer Architecture

The high-performance minicomputer forms the "heart" of the LDVT processor and accounts for most of the circuitry. To handle the anticipated rigorous real-time processing loads, the machine's architecture had to be sufficiently simple to accommodate maximum rate-cycle times, yet sophisticated enough to permit implementation of a substantially powerful instruction set. At a 55-nsec cycle time, it should be possible to execute a large variety of nontrivial operations in a single machine epoch.

The end result (Fig. IV-1) is a 2's complement, 16-bit, essentially fixed-point processor with software-controlled, extended-precision capability. The major subassemblies are a 512 × 16-bit high-speed RAM used exclusively for program data and constants (M_D), a separate 1k × 16-bit RAM strictly for executable code (M_P), a bused file comprised of four active registers (A, X, P, B), a versatile arithmetic/logic unit (ALU), and an input-output (I-O) system. In a typical operation, an operand selected from M_D and another selected from the register file are operated on in the ALU. The result is returned to the register file which can be loaded from or stored into M_D using the ALU as an intermediary, where appropriate.

Each of the four conceptual elements of the register file has special functions. The A-register is the primary machine accumulator, but also serves as a bootstrap buffer for code destined for loading into M_P. The X-register can be used as an ancillary accumulator, but serves mostly as an indexing component in M_D address calculation. The P-register is actually the machine program counter, hence supplying address information to M_P. Alteration or sequencing of P in response to program status is normally controlled automatically by special hardware. However, its inclusion in the register file facilitates status save/restore operations in subroutine and interrupt handling. The B-register is actually a pair of registers that serve as interface buffers for the I-O system. Peripheral in-out traffic handling and initial power-up bootstrapping are effected through this port.

The ALU (Fig. IV-2) is divided conceptually into halves, only one of which can be actuated at a given time. One half consists of the logic necessary to perform the fundamental add/subtract and Boolean operations. Provisions are made for several output scaling options via a selection matrix. Subordinate logic also is included to implement overflow detection and carry status preservation for programmed multiple precision.

The other half is a 16- × 16-bit multiplication element that forms a 32-bit signed product in 220 nsec. The design is a re-entrant/reclocked type consisting of two hardware iterations of Booth's 3-bit multiplier coding algorithm. Four machine cycles are necessary to perform the effective eight iterations required to produce a 32-bit product. Any one of four possible 16-bit multiplier outputs can be selected at a time for transmission to the A-register. They consist of the lower product half, upper half, and two shifted versions of the upper half. The lower half is always preserved for future retrieval in cases where the full 32-bit product is desired.

The R- and MOR-registers serve as intermediate buffers for the operands sourced from the register file and M_D, respectively. They are necessary due to the pipelined timing structure of the processor. The R-register serves in a secondary capacity as an input buffer for M_D during data store operations.

The I-O system consists of single, 16-bit input and output channels along with appropriate control. Each of the channels is further multiplexed to four subchannels. Simultaneous input and output may be active, but only one subchannel of each type can be accommodated at a time. Transactions can be conducted on a vector priority interrupt basis, or by using a simple programmed test for completion. Input takes priority over output and only one level of interrupt service routine nesting is permitted, i.e., once an interrupt has been honored, all further interrupts are locked out until the interrupt service is completed. Completion is signaled via a special indirect branch instruction used to terminate the service routine and to return to the main program. Overflow, ALU carry, and program counter statuses are saved automatically on interrupt. They are restored via the special termination instruction. Active register status must be saved and restored under software control.

2. Instruction Formats

To minimize cycle time, it was essential that a control with minimum decoding requirements be designed. For this reason, the LDVT minicomputer is virtually a one-format machine. The format (Fig. IV-3) consists of a 6-bit operation code field, a 9-bit address/constant field (y), and a single-bit special field (x). With the necessity of differentiating among several formats as a function of OP code eliminated, decoding could be effected efficiently by a fast 32- × 64-bit, micro-code ROM. Although the ROM technique affords the obvious advantage of custom instruction-set tailoring, its primary advantages are compactness and speed. The LDVT control constitutes somewhat of a degenerate case of the classic microprocessor control in that all but one machine instruction can be implemented in a single microstep. Overhead operations such as program counter maintenance and memory address calculation are performed automatically and in parallel with special explicit control logic.

The instruction repertoire that evolved, summarized in Table IV-1, can be classified in three basic categories according to the type of action governed. The first of these, the arithmetic/logic class, is of the general form:

$$f\{[R], [M_D(\alpha)]\} \rightarrow [R]$$

where

$$R = A,\ X,\ B,\ P$$

$$\alpha = \begin{cases} Y & , \quad \text{if } x = 0 \\ Y + [X] & , \quad \text{if } x = 1 \end{cases} .$$

The 9-bit y-field serves as a base address, capable of spanning all of M_D, which can be modified under control of the x-bit by the contents of the X-register.

The second class is the memory transfer group and has virtually the same structure as the arithmetic operations. Governed operations are of the general form:

$$[M_D(\alpha)] \rightarrow [R]$$

or

$$[R] \rightarrow [M_D(\alpha)]$$

where R = A, X, B, P as before. Operations of the form

$$[M_D(\alpha)] \rightarrow [P]$$

have the interesting effect of branching the running program. In fact, this is the means by which return-point restoration and indirect jumps are actually implemented.

The most interesting class is the control group and contains all conditional/unconditional branch codes as well as miscellaneous in/out handling instructions. Branches are of the general form

$$Y \rightarrow [P]$$

if conditions are met and

$$[P] + 1 \rightarrow [M_D(1)] \quad , \quad \text{if } x = 1 \quad .$$

Described verbally, a branch to location Y in M_P can be conditionally or unconditionally effected, and P status (return point) saved optionally in location 1 of M_D. Given a 1k M_P and a 9-bit y-field, branches can take place only within a 512-word page. Page boundaries are crossed using memory transfers into P, as described previously. Condition codes include overflow, input/output status, ALU sign, and sense switch tests. Auto-incrementing/decrementing jumps operating in conjunction with the X-register also are included.

3. Timing Philosophy

The following sequence of events must occur to fully execute a given instruction:

(a) P-counter assumes desired state
(b) M_P accessed
(c) Fetched instruction interpreted, decoded
(d) M_D address computed, if applicable
(e) M_D and register file read
(f) Execution
(g) Result recorded.

TABLE IV-1

LDVT INSTRUCTION LIST

Mnemonic	Action	Execution Time
LDA/LDAX	$[A] \leftarrow [M_D]$	T
LDB/LDBX	$[B] \leftarrow [M_D]$	T
LDP/LDPX	$[P] \leftarrow [M_D]$	T
LDX/LDXX	$[X] \leftarrow [M_D]$	T
STA/STAX	$[M_D] \leftarrow [A]$	T
STB/STBX	$[M_D] \leftarrow [B]$	T
STP/STPX	$[M_D] \leftarrow [P]$	T
STX/STXX	$[M_D] \leftarrow [X]$	
ADDA/ADDAX	$[A] + [M_D] \rightarrow [A]$	T
ADDP/ADDPX	$[P] + [M_D] \rightarrow [P]$	T
ADDX/ADDXX	$[X] + [M_D] \rightarrow [X]$	T
SUBA/SUBAX	$[A] - [M_D] \rightarrow [A]$	T
SUBP/SUBPX	$[P] - [M_D] \rightarrow [P]$	T
SUBX/SUBXX	$[X] - [M_D] \rightarrow [X]$	T
MULI/MULIX	Bits 0-15 of $[A] \times [M_D] \rightarrow [A]$	4T
MULF/MULFX	Bits 15-30 of $[A] \times [M_D] \rightarrow [A]$	4T
MULD/MULDX	Bits 14-29 of $[A] \times [M_D] \rightarrow [A]$	4T
MULH/MULHX	Bits 16-31 of $[A] \times [M_D] \rightarrow [A]$	4T
STPLA	Lower byte of last product $\rightarrow [A]$	T
AAND/AANDX	$[A] \cap [M_D] \rightarrow [A]$	T
AOR/AORX	$[A] \cup [M_D] \rightarrow [A]$	T
AXOR/AXORX	$[A] \otimes [M_D] \rightarrow [A]$	T
CMPA	$[\bar{A}] \rightarrow [A]$	T
ADDAD/ADDADX	$[A] + [M_D] + C_{save} \rightarrow [A]$	T
SUBAD/SUBADX	$[A] + [\bar{M}_D] + C_{save} \rightarrow [A]$	T

TABLE IV-1 (Continued)

Mnemonic	Action	Execution Time
LDAYP	$000000 + IR_{0-9} \rightarrow [A]$	T
LDAYN	$176000 + IR_{0-9} \rightarrow [A]$	T
DBA	$2 \cdot [A] \rightarrow [A]$	T
HVA	$2^{-1} \cdot [A] \rightarrow [A]$	T
QTA	$2^{-2} \cdot [A] \rightarrow [A]$	T
DBX	$2 \cdot [X] \rightarrow [X]$	T
HVX	$2^{-1} \cdot [X] \rightarrow [X]$	T
YIX	$Y \rightarrow [X]$	T
IOS	Initiate I/O transfers	T
STAMP/STAMPX	$[A] \rightarrow [M_P(Y + [X])]$	2T
JP/JPX/JPS/JPKS	$Y \rightarrow [P]$	T
JPZA/JPZAK/JPZAS/JPZAKS	$Y \rightarrow [P]$ if $[A] \geq 0$	T
JNA/JNAK/JNAS/JNAKS	$Y \rightarrow [P]$ if $[A] < 0$	T
JPZX/JPZXK	$Y \rightarrow [P]$ if $[X] \geq 0$, $[X] - 1 \rightarrow [X]$	T
JNX/JNXK	$Y \rightarrow [P]$ if $[X] < 0$, $[X] + 1 \rightarrow [X]$	T
JIR/JIRK/JIRS/JIRKS	$Y \rightarrow [P]$ if input transfer ready	T
JOR/JORK/JORS/JORKS	$Y \rightarrow [P]$ if output transfer ready	T
JOV/JOVK/JOVS/JOVKS	$Y \rightarrow [P]$ if overflow flag set	T
JSW/JSWK/JSWS/JSWKS	$Y \rightarrow [P]$ if sense switch W set	T
JSV/JSVK/JSVS/JSVKS	$Y \rightarrow [P]$ if sense switch V set	T
IJP	$[M_D(1)] + Y \rightarrow [P]$	T
IOIJP	$[M_D(2)] + Y \rightarrow [P]$	T
HLT	Stop execution	T

Notes:

T = 55 nsec

Suffix X appended to a mnemonic signifies that M_D address is Y + [X], otherwise it is Y + 0.

$[M_D(0)] = 0$. Thus, an "LDA 0" clears A, etc.

Suffix S appended to jump code signifies that return point is to be saved, i.e., $[P] + 1 \rightarrow [M_D(1)]$.

Suffix K appended to jump code signifies suppression of the next subsequent operation. Transfer time is effectively 2T in this case.

Machine NO-OP is a "STA 0."

Assuming the fastest circuit technology available, it would be impossible to accomplish this sequence in 50 nsec unless an utterly simplistic machine structure with very small memories is assumed. Calculations indicate that the above event chain could be segmented in thirds in a well-balanced way yielding a net cycle time on the order of 55 nsec. This implies a triple-overlapped, pipelined type of timing arrangement with the usual attendant increase in control complexity. However, experience shows that the overall package count increases suffered in such cases are usually modest and that the increased cycle time potential justifies the sacrifice.

To clarify details, consider the following symbolic code segment:

$[A] + [M_D] \rightarrow [A]$

$[A] \rightarrow [M_D]$

JPA Y .

In this example, the A-register is added to a location in M_D, the result is tested, and a branch to M_P (Y) takes place if it is positive. In a timing diagram of this sequence (Fig. IV-4), three time lines are marked off in units of machine cycles corresponding to M_P activity, decoding and setup, and final execution. The process begins by fetching the "add" instruction from M_P. At the end of the access cycle, the instruction is buffered in an instruction register (IR) and M_P is accessed again to fetch the "store" instruction. Simultaneous with the second access, the "add" instruction is decoded and the register file is read. Also, the M_D operand address is computed and M_D is read. At the instant the "store" instruction is loaded into the IR, the operands associated with the "add" instruction are loaded into ALU buffers R and MOR. During the next cycle, the "jump" instruction is fetched, the "store" instruction is decoded, and the "add" takes place in the ALU. At this point, the three-level pipeline is full.

M_D address calculation requires half a machine cycle (25 nsec). The actual read takes place during the latter half. Rather than leave the memory idle during the first half, it is available for store operations. Therefore, the execute portion of a store instruction actually occurs during the first half of the decode epoch of the subsequent operation.

A curiosity of the pipelined type of timing arrangement involves emptying the pipeline on a branch operation. Because of the overlap, a further instruction is read from M_P before the control realizes a branch is to occur. In essence, a cycle is needlessly lost in emptying the pipe. In the case of the LDVT minicomputer, it was decided that this cycle be available for use on an optional basis. That is, each branch instruction can either waste the cycle or not. If the cycle is used, the effect is to perform the next instruction subsequent to the branch, irregardless of whether the branch actually takes place. Many programmers find this an exceedingly useful, though somewhat unusual, feature.

4. Peripheral System

To make a self-sufficient speech terminal out of what has been described as a general-purpose minicomputer required a wholly integrated set of appropriate peripheral elements. The LDVT peripheral complex (Fig. IV-5) consists of a 12-bit, analog-to-digital/digital-to-analog converter (ADC/DAC) set, two 16-bit serial-to-parallel/parallel-to-serial converter (S-P/P-S) sets, 4k × 16 ROM, 2k × 16 RAM, and a host-computer channel.

The ADC/DAC set serves the obvious purpose of interfacing the local handset. The S-P/P-S sets mediate traffic flow of serialized data out to modems that interface with telephone lines or

whatever other transmission medium is desired. The two sets provided include a conferencing capability wherein a given LDVT can transmit from one speaker yet receive from two others.

The host-computer channel permits program assembling and editing in laboratory-based experimental environments. New software systems are thus transmitted easily to the LDVT. The host computer is also an effective debugging tool to monitor LDVT memory dumps, etc. For stand-alone applications, however, a 4k × 16-bit bootstrap ROM takes the place of the computer channel. In such cases, the ROM contains the necessary operational firmware to personalize the LDVT to whatever speech-compression algorithm the user desires. ROM contents are loaded into the minicomputer automatically on power-up controlled by a nonvolatile bootstrap loader in the first few M_P locations. This bootstrap loader also can acquire code from a host computer, if desired.

A high-speed 2k × 16 auxiliary RAM (M_X) in the peripheral complex enhances the rather limited memory capacity of the minicomputer. Read/write operations can be streamed at a 200-nsec rate because of the RAM's high-performance capability and the way its control is wedded to the computer in-out complex. Address information is supplied through the X-register. In a typical operational system, M_X is used to store speech buffers, coding/decoding tables, or perhaps executable code bound for loading in M_P. The latter could occur when the running program is too large to fit into M_P at once, thus necessitating real-time code overlays.

C. ENGINEERING CONSIDERATIONS: SYSTEM FABRICATION AND PACKAGING

The stringent performance and compactness requirements of the LDVT minicomputer restricted the choice of circuit technology to 10,000-series emitter-coupled logic (ECL 10k), a fully populated 2-nsec MSI family. The lower performance requirements of the peripheral system and outside-world compatibility considerations indicated that standard 7400-series TTL could be utilized safely. The minicomputer has 498 ECL packages, all but 12 of which are of the 16-pin DIP configuration. The remainder, used in the ALU, are 24-pin DIPs. 197 TTL 16-pin DIPs serve the peripheral complex along with a small analog board containing the DAC/ADC system, associated sampling/desampling filters, and miscellaneous audio amplification.

Given the brief development interval allotted, the entire LDVT, except the analog subsystem, was built with wire-wrap construction techniques. It is well known that ECL 10k with a 3-nsec rise time can be well controlled in a wire-wrap environment as long as proper care is taken in signal-path conditioning and DC power distribution. For example, signal paths must be terminated properly to control reflections, and loads must be constrained carefully in number and physical position to preserve waveform quality. The terminations, ranging typically from 50 to 150 ohms, pose a special problem in that they consume board space and increase dissipation. The usual practice in ECL systems is to provide a special −2-V termination voltage in addition to the standard −5.2-V supply to conserve power. Since the DC distribution system must exhibit very high capacitance and low inductance in the interests of noise-margin preservation, explicit strapping of a −2-V supply on a standard, single-voltage, wire-wrap board is an extremely dangerous practice. For this reason a special family of wire-wrap board, intended for use with ECL systems and currently commercially available, was developed by Lincoln Laboratory. Although essentially similar to standard 180-pack configurations, they differ in that a second, buried voltage plane is provided, along with proper decoupling capability, to handle the −2-V distribution. In spaces between the 16-pin DIP sockets, special 8-pin, single-inline (SIP) sockets accommodate Cermet termination resistor packs of compatible configuration.

The sockets connect directly to the buried −2-V plane. In the LDVT system, only two standard terminator SIP values were necessary: 100 and 150 ohms. By connecting pairs in parallel, values of 50, 60, and 75 ohms also could be achieved.

Four power supplies, supplying 225 W of real power for the LDVT, include: 40-A switching supply for the ECL −5.2-V, 10-A linear regulator for the −2-V ECL termination voltage, 9-A linear regulator for the TTL +5 V, and ±15-V supply for the analog equipment. Four 3-in., low-acoustic-noise fans at 50 cfm each provide forced air cooling.

The basic LDVT package (Fig. IV-6) fits in a 19- × 5- × 22-in. drawer, occupies about 1.25 cubic feet, and weighs 60 lb. A small outboard box houses the analog equipment and serves as a receptacle for the handset. The LDVT digital electronics, housed on four wire-wrap boards arranged in a stack (Fig. IV-7), open for access much as the pages of a book. Interboard connections are provided by controlled-impedance, flat ribbon cables running along the spine or "binding," obviating the need for a back plane. The bottom three boards are of the special ECL variety and comprise the minicomputer. The topmost board is a standard, single-voltage, 180-pack, wire-wrap board accommodating most of the peripheral system.

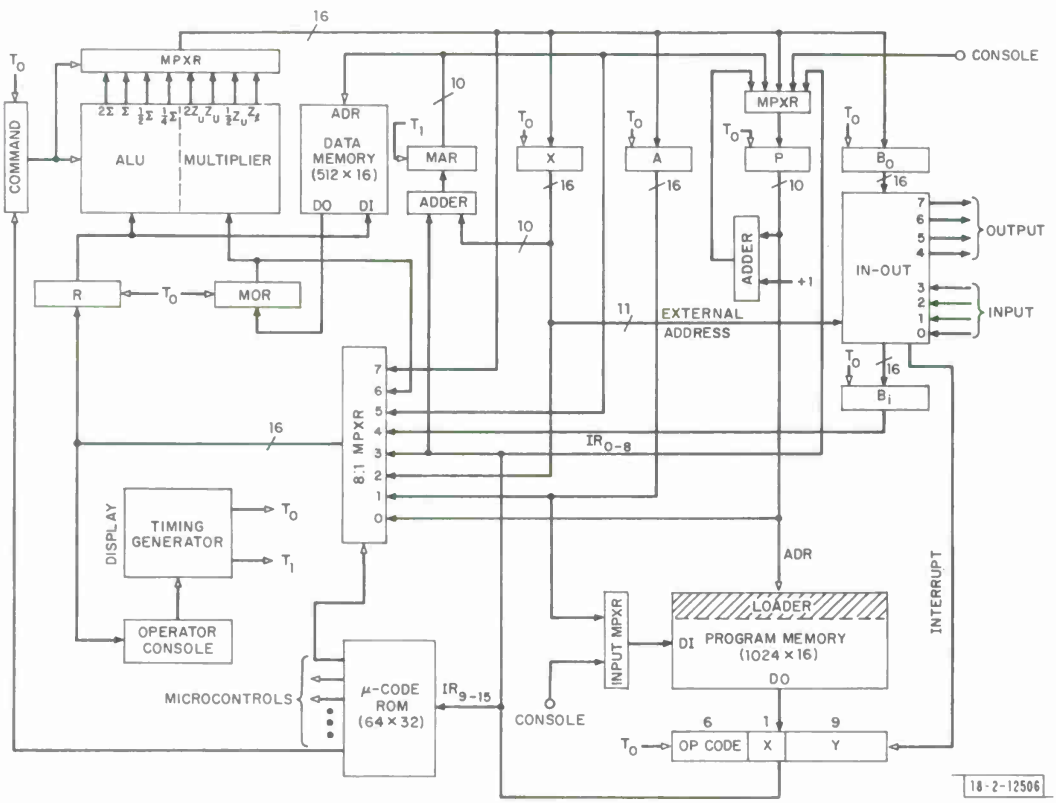


Fig. IV-1. LDVT minicomputer architecture.

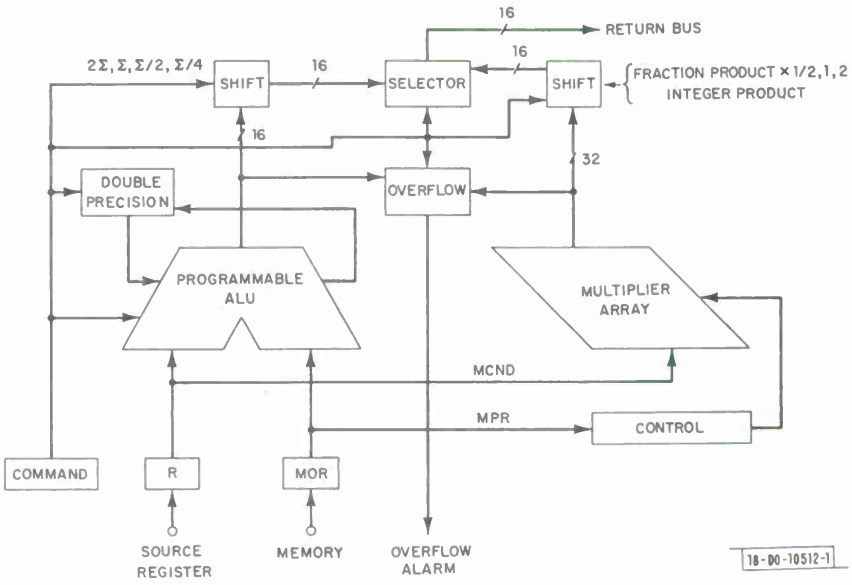


Fig. IV-2. LDVT arithmetic/logic unit (ALU).

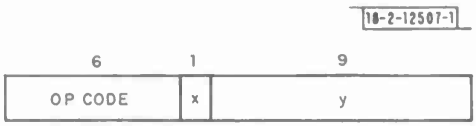


Fig. IV-3. Basic instruction format.

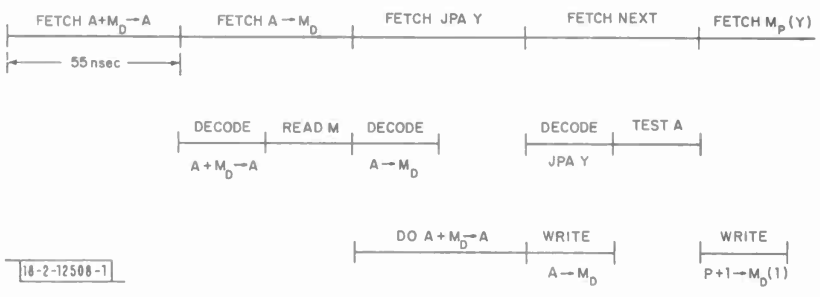


Fig. IV-4. LDVT timing example.

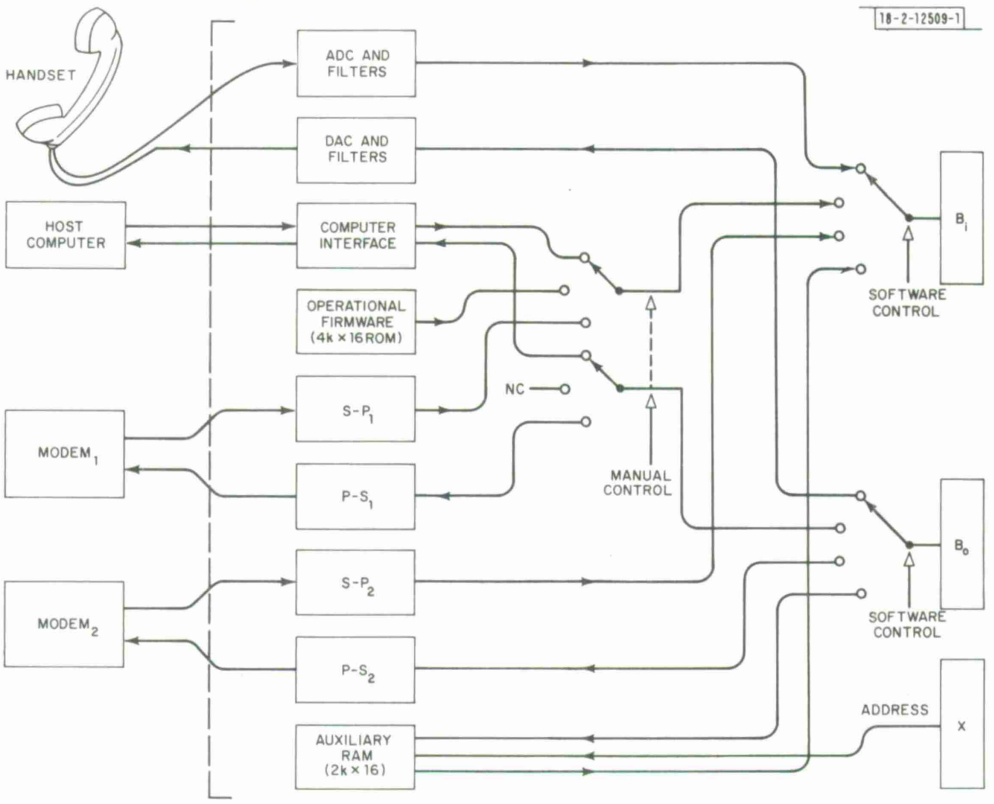


Fig. IV-5. Input-output complex.

-2-12649

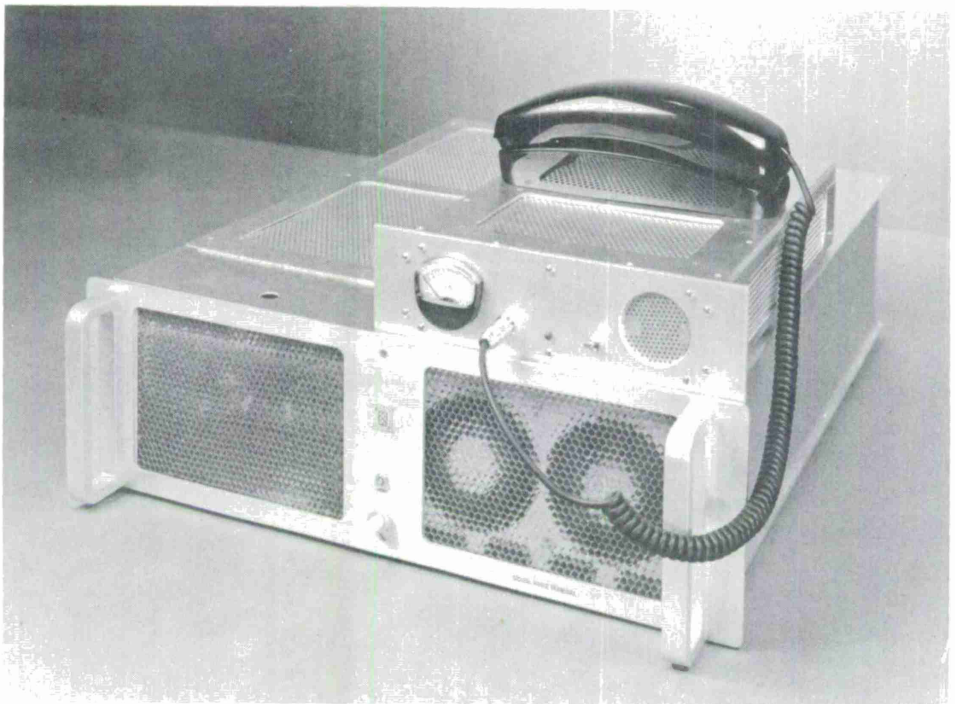


Fig. IV-6. LDVT ready for use.

-2-12650

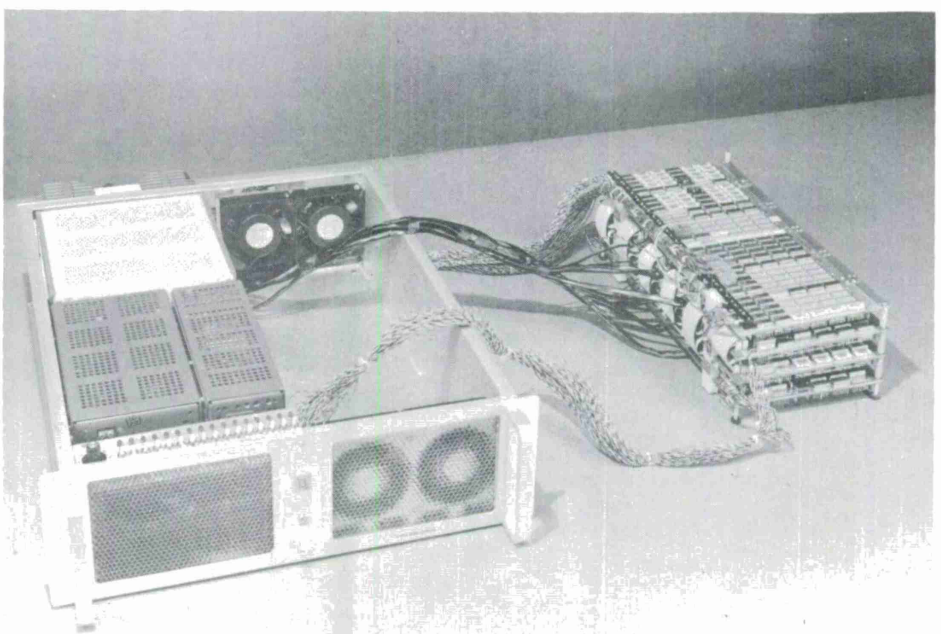


Fig. IV-7. LDVT open for service.

V. LARGE-SCALE INTEGRATION (LSI) MOTIVATED HARDWARE STUDIES

A. INTRODUCTION

It seemed worthwhile to study the problem of searching out innovative, efficient processor structures which take advantage of present-day technology evolutionary trends. The quest focused upon candidate designs which appeared promising from the following viewpoints:

(1) Low unit cost
(2) Amenable to high-volume production
(3) High reliability
(4) Compact form factor
(5) Flexible/versatile architecture.

Trade-offs in emphasis among the (potentially conflicting) desired objectives yield designs which can be roughly classified into three fundamental categories:

(1) Special Purpose:– This approach typically embodies the most efficient, compact, and inexpensive approach to implementing a particular choice of algorithm. The price that is paid, of course, is the relative inflexibility of the end product.

(2) General Purpose:– This class of processor, since it incorporates what basically amounts to a computer, is by virtue of its wholly programmable nature the ultimate in terms of flexibility. However, for a specific algorithm choice, inevitable inefficiencies imply a tougher overall performance requirement with all of the attendant problems indicative of high-speed technology system design. Stated simply, for a given problem the design is bigger and more costly than is probably necessary.

(3) Hybrid:– In the expansive middle ground lying between the aforementioned extremes, there exists a necessarily broad spectrum of designs which attempts to marry the best aspects of both worlds. Such hybrid designs are partly special purpose and partly programmable. For example, a functional building block common to many processing schemes (like correlation), but which is particularly taxing computationally, might be built as a special-purpose subsystem. But complicated specialized tasks, such as reflection coefficient extraction in an LPC vocoder analysis, might best be implemented in a limited programmable section.

It is our contention that a high premium should be placed on the more flexible design alternatives for active research applications areas such as speech processing. Given the many systems already in existence (APC, LPC, Channel, VELP, etc.) and the many more which will no doubt evolve, a fully flexible research vehicle seems essential. The first part of this report focuses upon the Lincoln Laboratory DVT. The intent is to suggest possible methods of reducing the cost and improving the form factor of the current design. Upon careful scrutiny the design is found to be dominated in terms of cost, integrated circuit (IC) count, and performance by its extremely fast memory complement. It is shown that little can be done to improve the design if constrained to maintaining the current performance levels with standard integrated circuitry.

It is further shown that a slower version which utilizes less expensive, more dense memory chips can be had at a 30-percent decrease in circuit count at a 50-percent speed penalty. Switching to a lower-speed technology is found to afford a similar package count reduction at a 100-percent penalty in speed, but the overall cost per unit is halved over the current design.

High-performance technology custom LSI is evaluated as an alternative to off-the-shelf parts. It is seen that this approach, which does not address itself to the memory area since that is considered a specialty, can be expected to impact little with respect to IC count on the current memory dominant design. Custom LSI also is found to be expensive in terms of developmental costs per unique part type. For low-volume production, such expenses cannot be justified.

A hybrid packaging scheme, wherein several dice of standard, off-the-shelf design share a common substrate, is suggested as a reasonable compromise approach. The developmental costs per part are about 1.5 orders of magnitude cheaper than LSI, and the memory density issue also can be accommodated. Form-factor and reliability improvements similar to those of genuine LSI can be expected, although the raw cost of IC dice as supplied by the vendors does not drop appreciably over that of standard packaged units.

The second part of this section concerns itself with the application of newly available bipolar LSI microprocessor chip sets to the problem of speech processor design. It is shown that the devices are, by themselves, far too slow to compete with DVT-like performance and that programmable parallel processing architectures based upon them do not yield satisfactory results in terms of utility, cost, form-factor improvement, or performance. Hybrid or quasi-programmable processor structures are suggested as likely application candidates for the microprocessors. One such structure, specialized to the task of LPC processing, is described. Initial estimates of IC count and attendant costs are indicated.

B. GENERAL-PURPOSE PROCESSOR CASE STUDY: THE DVT

To assess the DVT's performance in a practical situation, the essential software components of a 12th-order Markel LPC[1,*] vocoder system have been coded as a benchmark. The synthesis scheme, shown schematically in Fig. V-1, centers upon an all-pole time-varying filter as a model of the human vocal tract. The filter is excited by either a white noise source or a pulse generator controlled by the transmitted pitch period estimate, depending on whether a given frame is voiced or not. The more complex problem of analysis is shown schematically in Fig. V-2. Parameters characterizing the vocal-tract model for a given speech frame are extracted via an autocorrelation followed by a Levinson recursion.[2] Asynchronous pitch estimation is conducted in parallel using the Gold-Rabiner method.[3] The 12 filter parameters, voice energy level estimate, buzz/hiss decision, and pitch period estimate are finally encoded and packed for transmission.

Computation time estimates for the various requisite processing tasks are listed in Table V-1. Each task is categorized as to whether it belongs to analysis or synthesis, and whether it must be performed once per speech sample or once per frame. The table was compiled assuming a sampling rate of 6.6 kHz, and 22.5-msec speech frames overlapped by 33 percent which is equivalent to an intersample period of 150 μsec and an effective frame rate of 67 Hz. The autocorrelation time assumes double-precision arithmetic and that two correlation updates are performed on each sample arrival. Based on this information, the DVT is capable of exceeding real time by about 100 percent for this LPC implementation.

*Numbered references appear at the end of the text in each section.

TABLE V-1

MARKEL LPC-12 REAL-TIME PERFORMANCE

	Computation	T_{comp} (μsec) Per Sample	T_{comp} (μsec) Per Frame
Analysis	Correlation and window	20	–
	Filter parameter extraction	–	262
	Pitch determination and buzz/hiss decision	35	275
	Parameter encoding	–	88
Synthesis	Parameter decoding	–	13
	Buzz/hiss generation	1.6	–
	Filtering function	11.1	–
	Totals	67.7	638
	$T_{comp}/T_{avail} = \frac{67.7 + 638/100}{150} = 0.49$		

In order to assess what might be done to improve package count and cost, it is interesting to see how the DVT's nominal 470 ECL IC allotment and $13,000 outside purchase budget was spent. Table V-2 lists the programmable processor's major subassemblies and the emitter coupled logic (ECL) circuit count associated with each. A striking observation is that something over a third of the circuits were used up in the two internal memories. In terms of dollars, these two items comprise about two-thirds of the overall circuit cost for the programmable processor. Table V-3 summarizes these facts.

Table V-4 enumerates in some detail the recurrent outside purchase (OP) charges sustained by Lincoln Laboratory related to the production of a single DVT unit. These figures do not reflect overhead associated with design, fabrication, and debug of each unit. Total IC costs comprise about 42 percent of the total, with the ECL accounting for a full 28 percent. If the ECL memory alone is examined, it is seen that these circuits comprise nearly 20 percent of the total. It is also interesting to note that wire-wrap charges plus the requisite circuit panels, wire, terminations, and decoupling capacitors amount to 20 percent of the total – as much as the entire ECL circuit cost! These observations reflect the cost penalty associated with a high-performance wire-wrap system. If a commercial vendor were to implement the current design with a very modest production-level projection (≈100 to 1000 units), he would attempt to minimize his costs primarily by:

(1) Obtaining quantity discounts on digital and analog semiconductor components, and

(2) Using multilayer PC boards (~4 signal layers) instead of wire-wrap.

TABLE V-2
DVT ECL PACKAGE COUNT BREAKDOWN

Subsection	16 Pin	24 Pin
P-register	28	0
Instruction register	22	0
Control decoding	14	0
Input/output	45	0
Clock generator and console control	30	0
ALU	39	4
16 × 16 multiplier	44	8
M_D address control	29	0
R-register gating	23	0
1024 × 16 program memory	86	0
512 × 16 data memory	83	0
Miscellaneous	21	0
Total	464	12

TABLE V-3
DVT ECL MEMORY COST BREAKDOWN

Item	Count	Percent Count	Cost	Percent Cost
M_P	86	18	$1350	36
M_D	83	18	1000	27
Other	300	64	1250	37
Total	469	–	$3600	–

TABLE V-4
DVT SUBASSEMBLY COST BREAKDOWN

Item	Cost	Percent Cost
ECL circuits	$ 3,700	28
TTL circuits	1,800	14
Analog devices	700	5
Power supplies	1,000	8
Wire-wrap panels	2,000	16
Wire-wrap charges	600	5
Resistors/capacitors/wire	950	8
Connectors	700	5
Enclosures	650	5
Miscellaneous	1,000	8
Total recurrent OP costs per unit	$13,100	

Estimates indicate that, for a commercial DVT, the $13,100 figure (Table V-4) would drop to about $8200 given the above considerations.

Table V-5 suggests some minor design revisions which essentially retain current processor performance while permitting some small circuit count reductions. It is possible to eliminate a few circuits from the arithmetic section by removing some shift multiplexing and using a new multiplier chip which is due from Motorola in the first half of 1975. Some control revisions, such as register clock gating in lieu of recirculation, also could effect some savings. But, in all, a reduction of only about 50 circuits seems possible.

Clearly, in order to realize any appreciable package count and cost improvements it is necessary to attack the memory dominance issue. Memory densities increase and cost/bit decreases

TABLE V-5
REFINEMENTS OF CURRENT DESIGN

Design Revisions	ICs Saved
1. Use MC 10183 in DVT multiplier	17
2. Gate register clocks instead of recirculate	15
3. Use Hex D (10176) flip-flops and Hex (10195) inverters in clock generator	9
4. Four ALU output options instead of eight	9
Total	50

TABLE V-6
ALTERNATE DVT DESIGNS

Design Alternatives	Cycle Time	ICs Saved
1. Triple overlap with RAM M_P	55.0	50
2. Triple overlap with 1k × 40-bit ROM as M_P	55.0	96
3. Triple overlap with slow M_P (F10415) and M_D (F10410)	81.3	130
4. Double overlap with slow M_P and M_D	83.0	152

as performance requirements are relaxed. Table V-6 suggests some design revisions which take advantage of cheaper memory at a penalty in overall processor performance. Item 2 implies a resident nonvolatile program memory (ROM) and precludes operating the DVT in anything but a stand-alone mode. Items 3 and 4 retain the present random access memory structures in M_P and M_D, but use slower memory devices. As it happens, a minimal performance penalty is suffered in changing the processor's timing philosophy from a triple-overlapped to a double-overlapped arrangement while saving some additional control circuits. This can be seen by comparing the cycle times of items 3 and 4. The net result is that essentially the same processor structure can be retained while eliminating about one-third of the ICs at a performance penalty of 51 percent. Since the LPC-12 benchmark program appears to run at half real time, such a performance degradation would appear easily tolerable for this application at least. In terms of money, the ECL components cost would be reduced to about $2900 – an improvement of 20 percent.

If a factor-of-2 in performance degradation can be withstood, it seems reasonable to consider a technology shift to a saturating logic family such as the standard 54/7400 series TTL MSI. There is ample motivation for doing this since parts and fabrication costs can be drastically reduced. An improvement in form factor also can be expected because much more compact power supplies could be employed. (About half of the 1.25 ft^3 volume occupied by the DVT is power supply.) Calculations indicate that a TTL design corresponding to item 3 of Table V-5 would exhibit the same package count as the ECL version at an IC cost savings of about 50 percent. This is primarily due to the relatively inexpensive TTL memory chips. System design cost savings are realized also in such areas as circuit panels, terminations, power supplies, power-supply decoupling, and metal work. Rough calculations indicate that an overall fabrication cost savings of about 50 percent can reasonably be expected. However, a 110-nsec cycle time design is not possible with standard TTL. Upwards of 130 nsec is a more reasonable estimate. It would be necessary to make use of a limited number of judiciously selected high-speed TTL circuits (Schottky series) to attain a 110-nsec cycle time goal. This complicates the system design and increases the power budget somewhat, thereby compromising expected savings in these areas.

With the advent of several viable bipolar LSI technologies,[4] it is informative to consider their implications with respect to the current DVT design. In present-day terms, LSI implies 500 to 10,000 devices per chip. Some rather obvious advantages of this philosophy are:

(1) Minimum system size, weight, and power dissipation

(2) Fewest number of chips per design

(3) High reliability due to decreased number of IC interconnects

(4) Improved maintainability

(5) Improved performance potential due to minimized interconnect lengths

(6) For high volume production, recurring fabrication costs per unit are minimized.

The disadvantages are simply the high development cost and relatively long design cycle time per unique part type. Expenditures on the order of $50,000 to $100,000 per chip design, and turnaround times on the order of 9 to 12 months are not unusual.

To specify a custom family of LSI chips for the DVT with a minimum number of unique part types, the existing design must be partitioned in an optimum manner. In order to do this effectively, it is desirable that the design exhibit a regular or iterative topology. If it turns out that this is not the case, it is necessary to define very complicated, cumbersome chips to keep the number of part types under control. Such chips characteristically fall into what is termed "very large-scale integration" (VLSI) technology, which implies more than 10,000 devices per chip. Such complexity is beyond the present-day capabilities of ECL technology, but some work of this type has been done with the much lower performance emitter follower logic (EFL).[5] However, because of the decreased device performance of this technology, it does not seem possible to construct a DVT-like processor that can even meet real-time requirements, let alone match its performance.

Upon examination of the current design, it is seen that only the arithmetic and register file sections exhibit any apparent regularities. The very fertile area of memory is explicitly excluded, since no custom LSI house that we know of is doing work in this area. A 4-bit slice through the register file was considered, but pin out requirements imply a large header (at least 28 pins). Since only 10 percent of the total package count is tied up in this subsystem, LSI would have negligible overall effect here anyway. The adder/subtractor, using efficient MSI chips, is highly integrated already. The multiplier, however, could benefit from LSI both in local package count and performance potential, although the overall system form factor is not drastically improved. A 4- × 4-bit, 2's complement multiplier chip currently under development by Lincoln Laboratory, shown in Fig. V-3, is realized with a higher-than-standard performance ECL technology and can be packaged in a 28-pin header. Incorporated into the current DVT design, this chip would save twenty-five 16-pin packs and replace eight 24-pin packs with four of the 28-pin class. An attendant 25-percent improvement in multiplier performance also can be expected.

A less costly approach to form-factor improvement, which encompasses several of the benefits afforded by LSI and yet can be applied to the memory issue, is termed "hybrid packaging." With this method, standard die as supplied by the manufacturer are bonded to a common substrate. Chip interconnects are effected by wire bonds to single-layer substrate metalization. Performance, reliability, and even dissipation (due to reduced load capacitance seen by on-chip drivers) can be improved, not to mention a repairability feature. Development costs are on the order of a few thousand dollars per part type, and design cycle times are on the order of several weeks.

As a typical example, a 128- × 8-bit memory package, currently under development by at least one vendor, is shown in Fig. V-4. The design is based on a fast ECL 128- × 1-chip which accesses typically in 11 nsec. This particular configuration, containing 11 die and dissipating about 5 W, would substitute eight 28-pin packs for the 83-odd 16-pin packs which currently constitute M_D. A similar strategy could be formulated using the ECL 256- × 1-memory chip, yielding similar savings in the M_P design. Raw IC component costs do not improve with this technique, however, since manufacturers charge virtually the same for dice as for a packaged unit (based on charges for a molded plastic commercial header). But a real estate improvement of about 5:1 is realized in the M_D and M_P subsystems. Power dissipation density is certainly increased, but forced air coupled with miniature heat sinks still is a viable cooling approach.

From the foregoing discussion, the following conclusions are drawn with regard to the current DVT architecture:

(1) Given the degree of performance desired, the constraints of a standard package design, and a tight schedule, the choice of ECL technology in a wire-wrap environment was essential and the final package count, size, weight, and cost were not unreasonable.

(2) No significant improvement in package count, performance, and form factor is possible with currently available standard ECL integrated circuits.

(3) Significant package count reductions are possible only with a marked overall performance degradation. This is primarily due to constraints imposed by the bipolar memory dominance of the design in both cost and ICs. Use of higher-density memories which exhibit a lower cost/bit and concommitant performance degradation impacts heavily in both these areas.

(4) Switching to saturating logic technologies for low-performance options would cut overall costs in half and still yield a processor which is a factor-of-4 or -5 faster than those commercially available. However, it is not clear to us that processors of the DVT architectural ilk in this performance class are of high prospective utility as speech research tools, given the uncertainty in complexity and computational onus of future processing schemes.

(5) Due to the nature of the DVT architecture and the performance level demanded, it does not seem possible to define a small number of unique LSI parts, with complexities not beyond the realm of ECL technology, which would have more than a token impact on system IC count. Given the high development cost/part type and the relatively low level of DVT production expected, custom LSI should probably be rejected as economically unfeasible.

(6) The hybrid packaging approach does seem to exhibit a potential for overall system form-factor improvement, even in the memory area. Although apparently no dollars are saved in IC die procurement, the development costs/part type are at least an order of magnitude more palatable than the

LSI approach. However, the recurrent fabrication costs per piece may prove to be prohibitive since this is a very laborious process. Therefore, the hybrid technique should be investigated further, but cautiously, for memory dominant, low-production volume designs such as the current DVT.

C. QUASI-PROGRAMMABLE PROCESSORS USING BIPOLAR MICROPROCESSOR ELEMENTS

Within the last year, two relatively low-cost bipolar microprocessor chip sets have become available as standard offerings, and it appears that at least two additional manufacturers will be entering the marketplace in the near future. These circuits legitimately qualify in complexity as being of the LSI class, and are realized with a form of Schottky TTL technology. Applications areas which can withstand the performance limitations inherent in such devices can avail themselves of the following obvious advantages:

(1) As was shown earlier, a TTL system design is on the whole cheaper and less complex than a high-performance ECL system.

(2) LSI componentry affords many advantages, yet the exorbitant cost of devising custom parts for a particular design is avoided.

The LSI units described here impact greatly on what would normally be considered the arithmetic and control portions of a standard minicomputer architecture. They rely heavily upon recent advances in bipolar read-only-memory (ROM) manufacturing technology, and do not address the issue of random-access memory at all.

These chip sets are designed to be used in the context of a micro-programmed architecture,[6] a typical form of which is shown in Fig. V-5. The advantage of the micro-programming concept is that the character of the processor (i.e., the effective instruction set) is defined by the contents of a ROM. Therefore, a single general logic structure can, if fast enough, be made to look like (or emulate) any existing computer design from the user's viewpoint. The canonic architecture consists of a central processing element (CPE), a control, an input/output section, and a main random-access store for both code and data. The cleverness of this arrangement is embodied in the control, which is comprised of sequencing logic and the characteristic ROM. Each complex computer instruction is decomposed into a sequence of elemental steps (micro-instructions) which are contained in the ROM. The micro-program controller sees to it that each micro-instruction is executed properly in sequence and that new complex (or "macro") instructions are fetched from the main store at appropriate times. In actuality, the ROM is the key element in the design since it replaces much of the bothersome random logic characteristic of computer controls.

Block diagrams depicting the essentials of the two existing CPE elements are shown in Figs. V-6 and V-7. The unit of Fig. V-6 consists of a 2-bit slice through an arithmetic/logic unit (ALU), an 11-deep scratchpad register file, an accumulator, and an auxiliary buffer register.[7] Attendant decoding and selection logic also is provided locally on the chip. In a 16-bit context, this element is capable of 120-nsec clocking epochs for elemental micro-instructions such as an addition involving the accumulator and the scratchpad file. However, to perform a typical

macro-instruction, several elemental cycles may be required. A typical sequence for an addition between a scratchpad register and a location in main memory might proceed as follows:

(1) Compute effective memory address and store in address register.
(2) Load memory into accumulator (AC).
(3) Add scratchpad register to AC and store.
(4) Increment program counter and load address register.
(5) Load next macro-instruction into instruction register from main memory.

Thus, five elemental epochs are necessary to perform one macro-instruction and fetch the next, a total time of $5 \times 120 = 600$ nsec. This is about a factor-of-10 slower than the DVT. More complex operations such as multiplication can, unless special hardware is added, take up to 20 times longer than the DVT. Given that the architecture of this CPE is not terribly dissimilar to that of the DVT, it seems apparent that even two such microprocessors operating in parallel (one for analysis, one for synthesis) cannot vaguely approach the performance levels of the DVT for LPC.

For completeness, a second type of CPE is shown in Fig. V-7. It consists of a 4-bit slice through an ALU, a 16-deep 2-address register file, and an auxiliary register. Local decoding and selection logic is included. In a 16-bit context, this unit is capable of a 200-nsec μ-cycle epoch. Although apparently slower than the other CPE element, this unit features architectural advantages which could, in some applications, offset its relative sluggishness. The 2-address register file could reduce main memory accesses, thereby speeding up overall execution times. To test this thesis, the Levinson recursion portion of the DVT's LPC analyzer was coded on a paper-design processor based on this CPE. The design of interest employed much auxiliary external logic to reduce the number of μ-cycles per macro-op to the bare minimum (namely 1). Even so, the execution time turned out to be no better than the ratio of its clocking epoch to the DVT's. Hence, it was concluded that the 2-address cache memory does not afford any obvious advantages in this case, and the overall performance of this CPE could be expected to be even worse than that of the other for a full LPC. Another disadvantage of this element is that it is the only member of its chip set. The set which complements the 1-address CPE contains a micro-controller, look-ahead carry block, and priority interrupt in/out control.

Returning for a moment to the notion of paralleling microprocessors to achieve performance equivalent to the DVT's, it is interesting to pose the question: Where is the point of diminishing returns? This query can be dealt with summarily by considering the case of four parallel 1-address processors sharing, perhaps, a common main store. The following conclusions can be drawn from studying such an arrangement:

(1) Although as general as the DVT, this is a far more difficult structure to coordinate and program.

(2) In terms of performance this arrangement is still, on the average, $11/4 = 2.75$ slower than the DVT.

(3) At current pricing levels, a stripped 16-bit microprocessor (exclusive of random-access memory) costs about $800 in small quantities, including some I/O control. Therefore, the proposed arrangement will cost about $3200 in circuits with main memory yet to be added! From this result, it seems far more advisable to build a 110-nsec DVT in TTL MSI

which is known to be a far cheaper expedient and certainly an easier architecture to use.

(4) In terms of IC count, each 16-bit elemental processor requires about 25 chips. Including main memory, the entire structure can be expected to require around 150 to 200 chips. However, many of the chips are 28- and 42-pin configurations. Hence, overall real estate savings are not improved as much as might be thought over a 300-can TTL realization of the standard DVT architecture.

A fully general structure, consisting of several parallel microprocessors, seems to be a losing proposition in terms of utility, cost, complexity, performance, and form-factor improvement. A better approach is to consider a somewhat specialized structure which retains a fair degree of flexibility through programmability. As an example, a processing structure based on the Markel LPC class of algorithms and employing two microprocessors is shown in Fig. V-8. The upper portion addresses the task of analysis. Straightforward real-time correlations are performed using special-purpose digital hardware. But the less taxing (though conceptually more sophisticated) jobs of extracting filter parameters, coding/formatting information, and I/O supervision are programmed in a microprocessor. The pitch extraction path also is done partially in special-purpose (analog) hardware and partially in the microprocessor. In the synthesis section, I/O supervision, decoding, buzz/hiss generation, and vocal-tract filter computations are done in the second microprocessor. A random-access memory complex supplies code and working storage space to each microprocessor. Some of this, such as the encoding/decoding tables, could be common storage. The program memories should be independent, however.

Expected performance can be inferred from Table V-1. Under synthesis, it is seen that the DVT uses up about 13 of a 150-μsec budget. A processor on the order of 10 times slower than the DVT doing only synthesis might use up 130 μsec. It would seem that a comfortable margin relative to the 150-μsec constraint is therefore maintained. In the analysis section, the tasks of correlation and most of the real-time pitch analysis are done in external special-purpose equipment. The remaining jobs need only be done once per frame, implying that a processor 10 times slower than the DVT would have no real-time problems if confined to only these tasks. Thus, it could perform other control tasks if desired.

The prospective IC count for such a structure does not seem unattractive either. Assuming a total random-access memory capacity of 2048×16, two 16-bit microprocessors, and miscellaneous circuitry for the correlator and input/output traffic, a total count of well under 200 chips seems possible. The IC cost would be in the range $2500 to $3000. It must be realized that these figures are very tentative and very preliminary. Although promising, much more intensive, detailed studies of this class of microcomputer-based architecture must be conducted.

D. SUMMARY AND CONCLUSIONS

In this section, we have shown that the Lincoln Laboratory DVT design as it stands represents a very creditable set of trade-off compromises when cost, size, performance, and utility are considered. The design was seen to be memory dominated in cost and IC count, and as such could not be expected to benefit much from a custom LSI technology which did not address this issue. The irregularity of DVT structure implies definition of several unique LSI part types which, because of the high developmental cost per part type, serves to further discourage any

TABLE V-7

SUMMARY OF DESIGN ALTERNATIVES

Design	Technology	IC Count	Size (ft^3)	Power (W)	Speed (norm. to real time)	Recurrent Lincoln OP Costs	Recurrent Commercial Vendor Costs
Current DVT*	ECL MSI TTL MSI	476 213	1.25	225	2.0	$13,100	$8,200
Double-overlap DVT (slower memories)	ECL MSI TTL MSI	324 213	1.25	175	1.29	$11,030	$6,745
Current DVT in TTL	TTL MSI	559	0.8	180	~1.0	$ 7,450	$5,837
Semiprogrammable using twin Intel microprocessors	TTL MSI/LSI	<200	0.3	<50	~1.0	$ 6,550†	$3,000

* Including all peripherals.

† Assumes bipolar memory.

more thoughts along this line. A hybrid packaging scheme is directly applicable to the memory problem, as well as to the rest of the miscellaneous logic comprising the machine, and at a much more tractable cost level. We feel that this is the best route to cost and form-factor improvement at the present DVT performance levels.

It was also seen that the new bipolar microprocessor chips by themselves yield results which are attractive in neither cost/performance nor package count. For a fully programmable structure, a standard TTL MSI copy of the ECL DVT is a more effective approach. However, semi-programmable processor designs, addressing specific algorithm classes (such as LPC), may represent viable cost/performance alternatives with significant form-factor improvement.

The four major design alternatives treated in the text are summarized in Table V-7. The first three entries may be compared and contrasted as DVT-like structures starting with the current design and ending with a low-performance, all-TTL copy of the ECL realization. It is also interesting to compare the last two entries, although not identical architectures, since they are both TTL systems. Two cost figures are given for each: the first represents an estimate of the recurrent Lincoln OP charges per unit (like Table V-4); the second is an estimate of what similar costs might be for a commercial vendor. It is seen that a commercial ECL DVT represents a very excellent buy if a flexible research tool is desired. However, for low-performance, high-production-level applications, the microprocessor structure looks most attractive. Given the usual market pressures that come into play as new microprocessors become available, the cost projections can be expected to drop further. It would seem that the commercial marketplace is, for our purposes, the best mechanism for solving the cost problems of LSI yet reaping the obvious advantages.

REFERENCES

1. B. S. Atal and S. L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave," J. Acoust. Soc. Am. 50, 637 (1971).
2. J. D. Markel, "Formant Trajectory Estimation from a Linear Least-Squares Inverse Filter Formulation," Speech Communications Research Laboratory, Santa Barbara, California, SCRL Monograph 7 (October 1971).
3. B. Gold and L. R. Rabiner, "Parallel Processing Techniques for Estimating Pitch Periods of Speech in the Time Domain," J. Acoust. Soc. Am. 46, 442 (1969).
4. B. Dunbridge, "LSI-Technology Overview," TRW Corporation, Redondo Beach, California, DCA Presentation, 10 July 1973.
5. H. S. E. Tsou, T. C. Berg, and S. E. Hutchins, "High Speed LSI Processor for Voice Signal Processing," TRW Systems Group, Redondo Beach, California (10 July 1973).
6. S. S. Husson, Microprocessing Principles and Practice (Prentice-Hall, New York, 1970).
7. J. Rattner, J-C. Cornet, and M. E. Hoff, "Bipolar LSI Computing Elements Usher in New Era of Digital Design," in Electronics Magazine (5 September 1974).

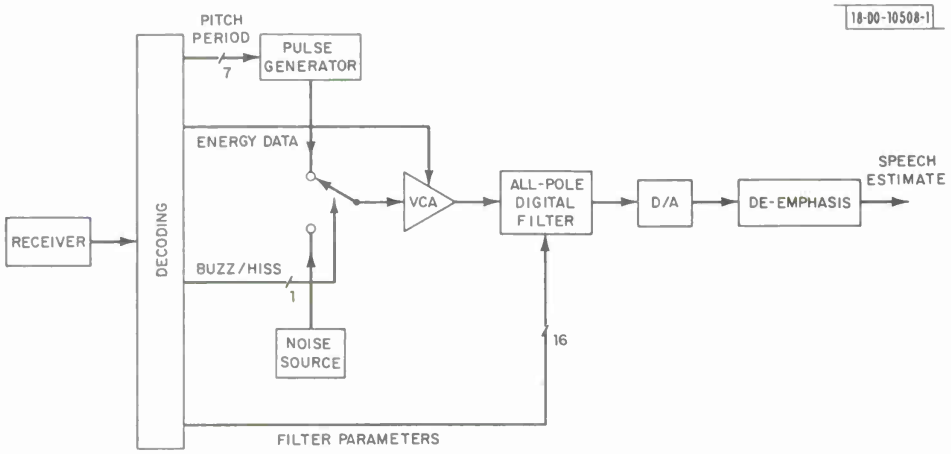


Fig. V-1. LPC-12 synthesis.

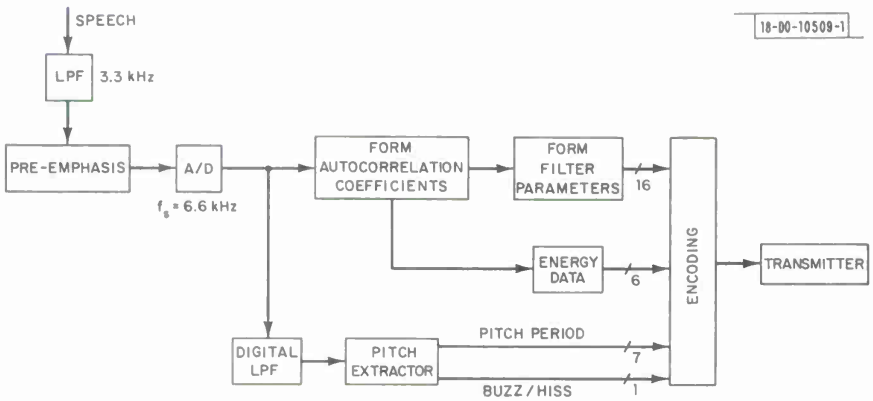


Fig. V-2. LPC-12 analysis.

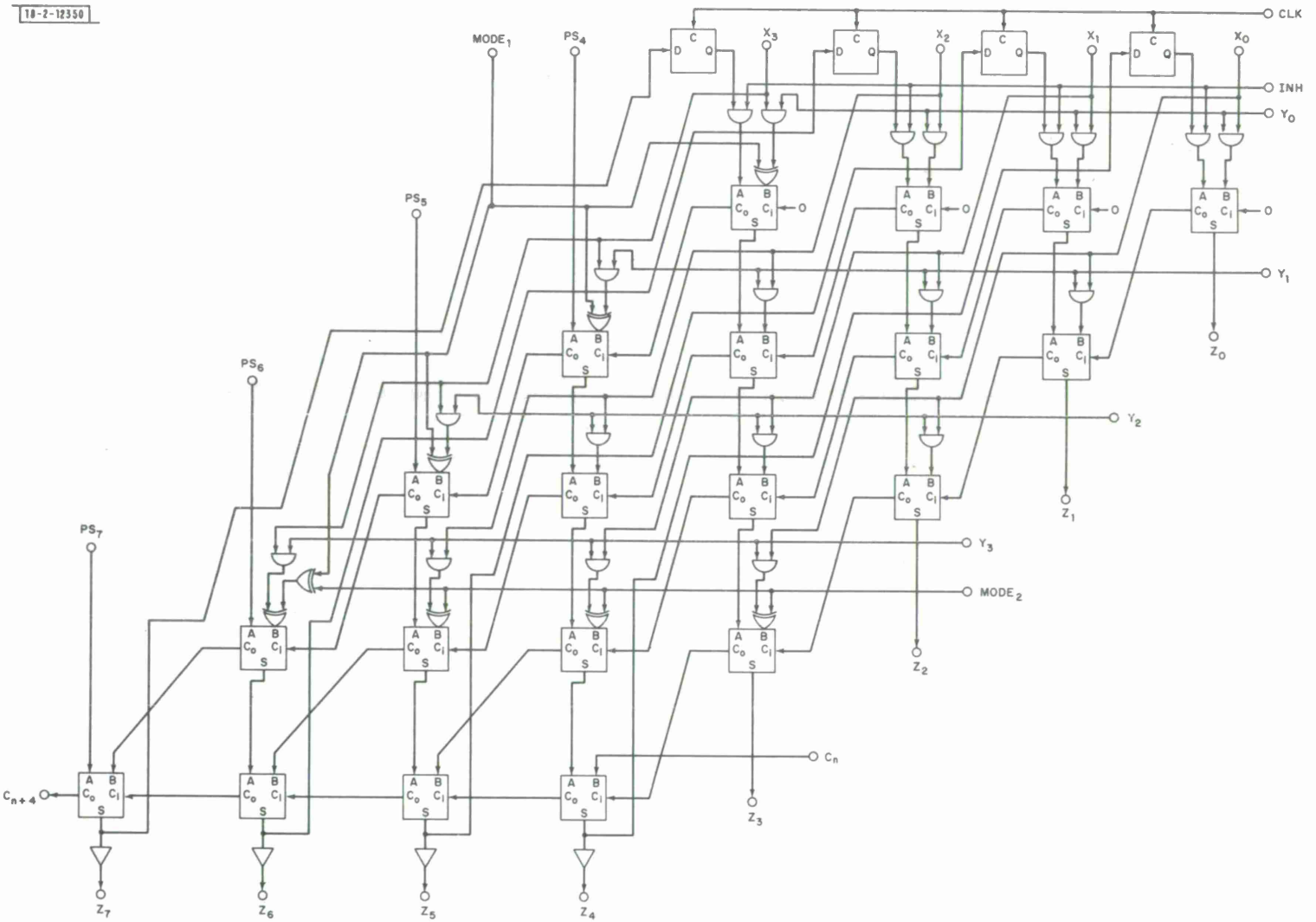


Fig. V-3. 4- × 4-bit 2's complement multiplier.

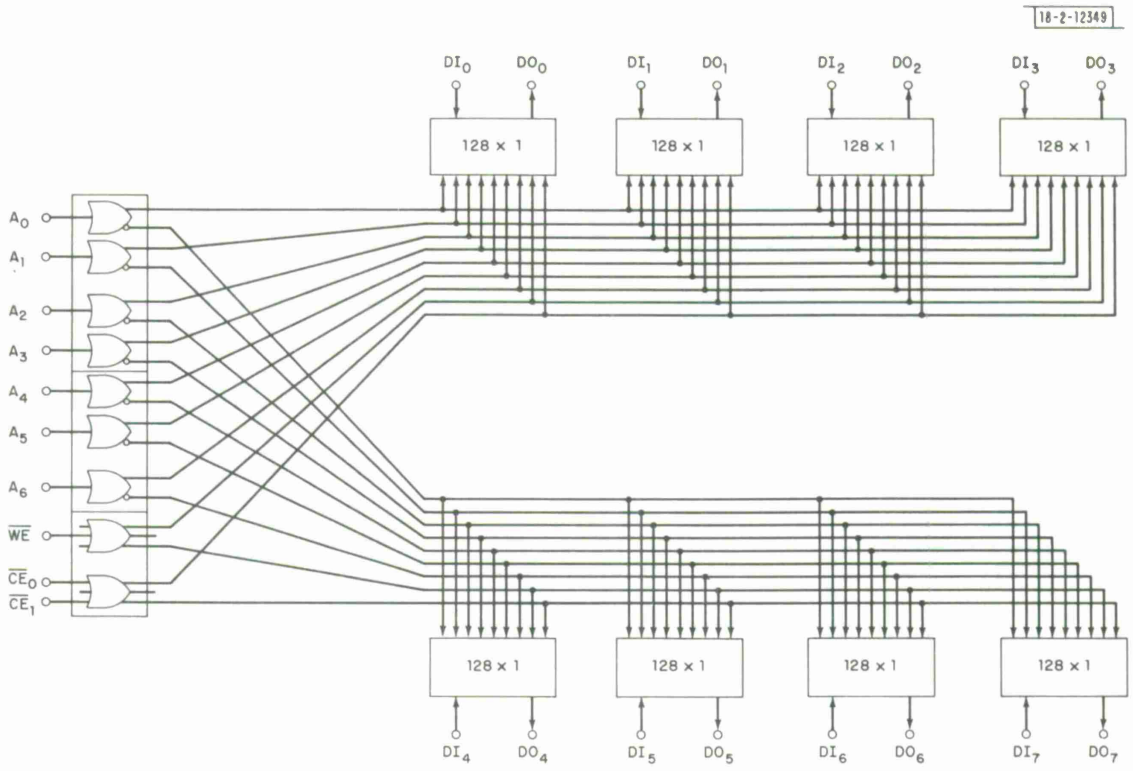


Fig. V-4. 128- × 8-bit ECL memory hybrid pack.

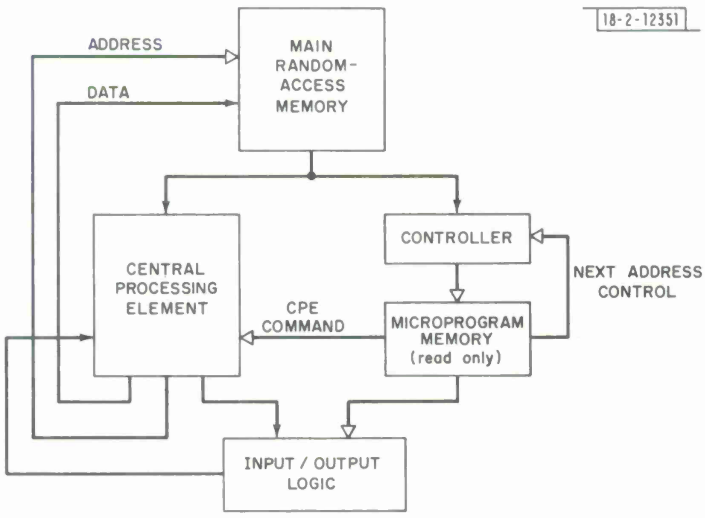


Fig. V-5. Typical microprocessor architecture.

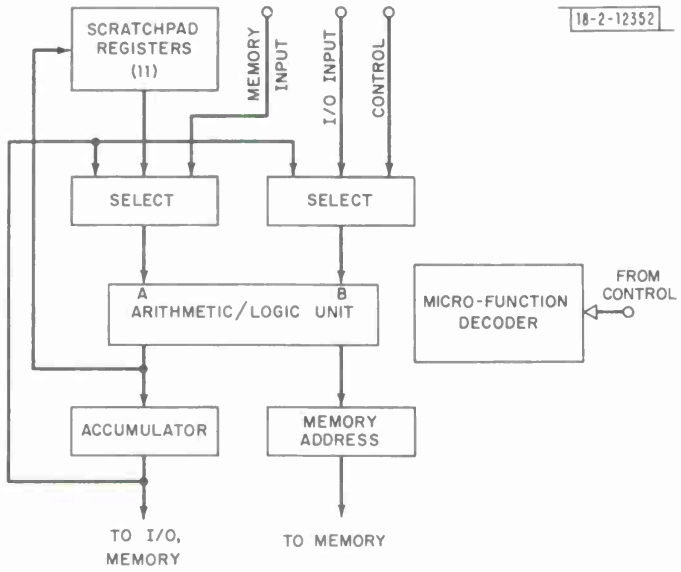


Fig. V-6. Single-address register file CPE chip.

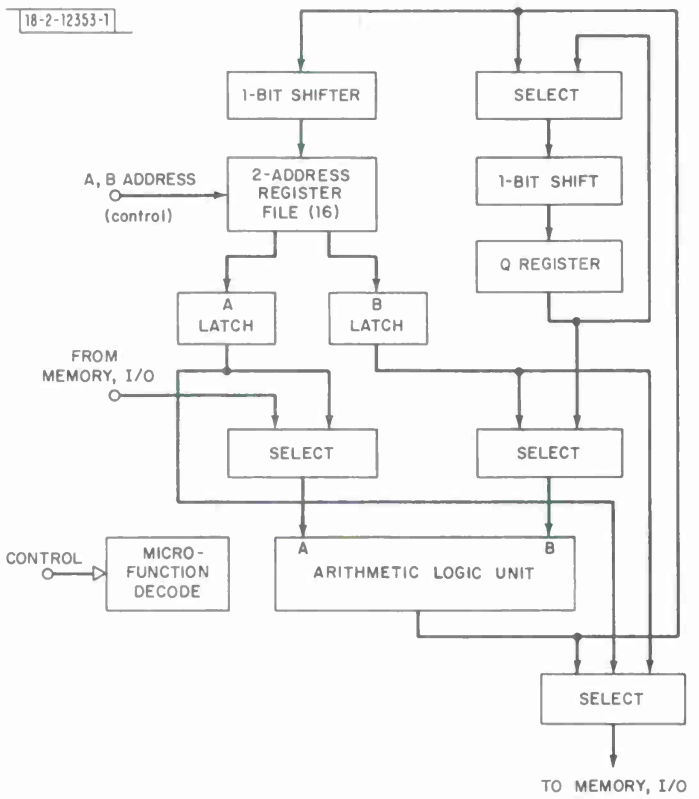


Fig. V-7. 2-address register file CPE chip.

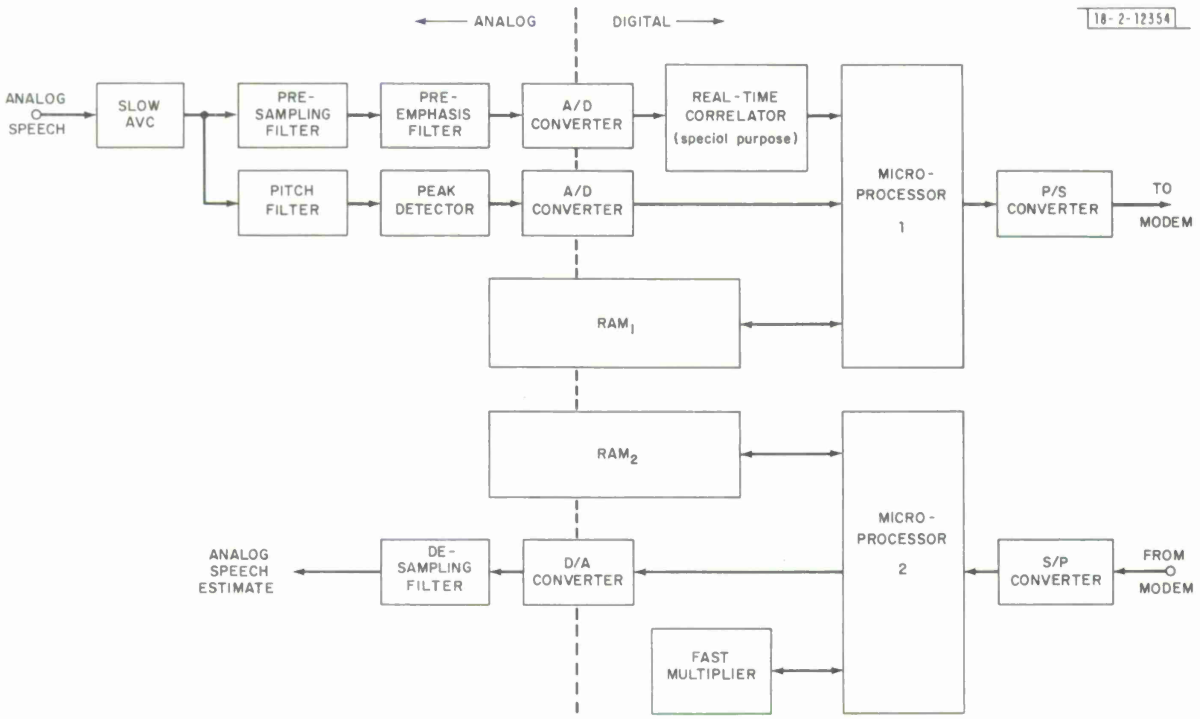


Fig. V-8. Semiprogrammable speech processor.

VI. ADAPTIVE RESIDUAL CODING STUDIES

Details of the Adaptive Residual Coder (ARC) are discussed in Ref. 1. The algorithm consists of a second-order fixed predictor and an adaptive error quantizer shown in Fig. VI-1. Two versions of the ARC have been realized – one with a 5-level quantizer for a transmission rate of 9600 bps, the other with a 7-level quantizer for a transmission rate of 16,000 bps. The adaptive quantizer has both a slow and a fast decaying memory of previous quantization levels, and it determines the unit of quantization. At the k^{th} instant, both the transmitter and receiver update the unit of quantization T(k), thus:

$$G'(k) = G'(k-1) \times \alpha + f_1[d(k-1)] \geq 0$$

$$C(k) = C(k-1) \times \beta + f_2[d(k-1)]$$

$$G(k) = G'(k) + C(k) + GMIN$$

$$T(k) = 2^{G(k)}$$

where $\alpha = (1 - 2^{-7})$, $\beta = (1 - 2^{-2})$, f_1 and f_2 are functions of the previous slice level d(k – 1), and GMIN is a constant. The quantity G'(k) serves to adjust the quantization unit based upon the long-term behavior of the slice levels. C(k) responds quickly to occurrences of outer slice levels, but decays more rapidly. Once the quantization unit has been computed, the quantized error Q[e(k)] and the slice level d(k) are determined as shown in Fig. VI-2. The predicted signal at the transmitter plus the quantized error are remembered for later use by the fixed predictor. The receiver adds the quantized error to its predicted value to produce the output signal which also is to be used by its fixed predictor.

In January 1975, in response to a request by DCA to assist G. D. Forney of Codex Corporation, several ARCs had been incorporated into a real-time facility on the FDP with a control program resident in the Univac 1219 which enabled instantaneous switching from one version of ARC to another. Both the 9.6- and 16-kbps systems were implemented, the versions differing in the following parameters: predictor coefficients, quantization unit factors, functions of the slice level, decay factors α and β, and GMIN. These systems ran back-to-back in the FDP, processing speech from audio tape or a handset. The different systems were evaluated, and preferable sets of parameters were decided upon by members of Codex Corporation. Because the ultimate ARC system was to be fully duplexed, involving variable length coding of the slice levels and buffering of these codes at the transmitter and receiver, further design was necessary. Further testing tried updating a bit count every sample by adding the number of bits needed to code the slice level and subtracting the average number of bits per sample as determined by the bit rate. This simulated the behavior of an actual transmitter buffer. By varying the bit rate to prevent buffer overflow, the various systems were observed to operate satisfactorily at slightly different bit rates. Codex Corporation was informed of these observations.

Using a set of parameters suggested by Codex Corporation, we then implemented a fully duplexed ARC system on the Lincoln DVT.[2] Speech is sampled and output at 165-μsec intervals via direct interrogation of the A/D and D/A converters, all processing occurring in real time. When the transmitter determines the slice level, this level is coded into one of five or seven variable length codes and entered into a 512-bit serial bit stream buffer. The receiver extracts and decodes the next slice level in its 512-bit buffer. A sufficient number of interrogations to the serial-to-parallel and parallel-to-serial converters is made during each 165-μsec interval

to assure that a modem clock pulse will never be missed. To prevent overflow and underflow of either buffer, a bit count is maintained at both the transmitter and receiver. Appropriate action is taken at the transmitter and receiver upon impending buffer overflow or underflow in order to maintain work synchronization while sacrificing speech quality. Channel errors, which may or may not cause loss of word synchronization, will result in degradation of the output speech; but the predictors and the quantizers at the transmitter and receiver will decay during silence, and the synchronization of transmitter and receiver states should be restored.

REFERENCES

1. S. U. H. Qureshi and G. D. Forney, "Adaptive Residual Coder – An Experimental 9.6/16 kb/s Speech Digitizer," EASCON 75, IEEE Electronics and Aerospace Systems Convention, Washington, D. C., 29 September – 1 October 1975.
2. E. M. Hofstetter, P. E. Blankenship, M. L. Malpass, and S. Seneff, "Vocoder Implementation on the Lincoln Digital Voice Terminal," EASCON 75, IEEE Electronics and Aerospace Systems Convention, Washington, D. C., 29 September – 1 October 1975.

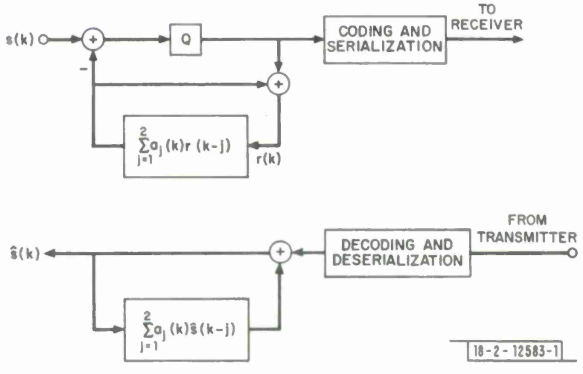


Fig. VI-1. The ARC vocoder.

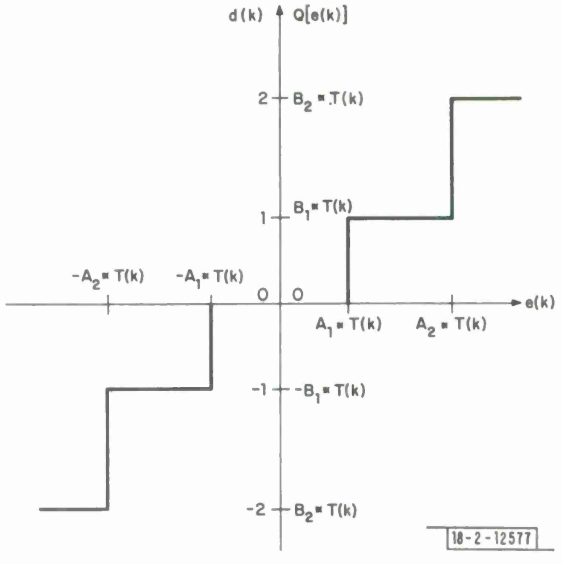


Fig. VI-2. The ARC quantizer.

VII. ADAPTIVE PREDICTIVE CODING (APC) STUDIES

A. INTRODUCTION

An 8-kbit APC algorithm was developed on the FDP and implemented on the DVT in real time. Certain experiments were conducted to optimize the quality at 8 kbits, using the FDP program, and the DVT implementation was chosen from among the various FDP versions as the one having the best quality for the least amount of complexity.

The final algorithm decided upon closely follows the algorithm described by Goldberg,* except that the sampling rate (and therefore the bit rate) is somewhat higher. Included is a fourth-order linear prediction, a pitch prediction, and a nonlinear feedback loop.

B. ALGORITHM REFINEMENT

Two major experiments were conducted on the FDP to optimize the APC algorithm. In the original Goldberg algorithm, pitch effects are first removed from the input speech waveform and then the linear prediction coefficients are determined from the resulting waveform. The RMS error of the LPC prediction is used to determine the gain parameter q. Since standard LPC algorithms determine the LP parameters from the original speech signal (without benefit of removal of pitch) and since removal of the vocal-tract filtering should help the pitch predictor, it could be an improvement to reverse the Goldberg algorithm by removing the vocal-tract effects before removing the pitch. Since the parameter q could no longer be determined from the LP error, it was determined instead as $1/N\ E|e(n)|$, where e(n) is the remaining error signal after both the vocal-tract and pitch effects have been removed. The results of this "inverse APC" algorithm were disappointing in that the quality of the input speech was essentially the same as in the original algorithm, while the complexity was greater.

The second experiment tried was to vary the cutoff frequency of the analog filter of the input speech. The speech is sampled at 150 μsec in the FDP version, and filters were built at 150, 170, and 190 μsec (3333-, 2940-, and 2630-Hz cutoff). It was found that by slightly oversampling, background noise was reduced, but with too much oversampling (190 μsec filter), the speech became overly muffled due to the low cutoff frequency of the filter. Therefore, 170 μsec was chosen as the optimal input filter.

C. THE CURRENT ALGORITHM AT 8 kbps

The APC algorithm is diagrammed in Fig. VII-1. The speech is filtered using a 170-μsec analog filter, i.e., the beginning of a 50-dB stop band starts at $1/2 \times 170$ μsec and sampled at 154-μsec intervals. Processing is begun on a new frame every 25.8 msec (N = 168 samples). The first step is to determine the pitch, M, using the simple (but time consuming) AMDF pitch detector.* After M has been determined (regardless of whether a frame is voiced or unvoiced), the pitch predictor coefficient α is computed. Both M and α are computed using double-precision arithmetic. It is now a simple matter to determine the waveform d(n) which is the output of the pitch filter and the input to the LPC analysis. In this analysis, five autocorrelation coefficients R_i are computed double-precision, but stored single-precision in a block floating point representation as in LPC. The linear prediction coefficients a_i, reflection coefficients K_i,

---

*A. J. Goldberg and H. Shaffer, "Low Data Rate Voice-Communication Using Small Computers," Proceedings of the IEEE Communication Systems and Technology Conference, Dallas, Texas, April 1974.

and residual error E are computed using a fourth-order Levinson Recursion. The parameter q is determined from the residual error by the empirical approximation, $q \simeq 0.72 \sqrt{E/N}$. Translation from E to q is achieved by table lookup in a 5-bit log table, to avoid the necessity of a square-root algorithm. The four reflection coefficients, M, α, and q all are coded and the bits are packed and stored in the modem transmit buffer. The reflection coefficients and α are coded using a 5-bit arc sine table, q is coded logarithmically, and M is left uncoded. The coded values of the parameters are used by the analyzer in the feedback loop shown in Fig. VII-1 to generate synthetic speech s(n), and the error signal e(n).

A direct-form filter, as contrasted with an acoustic-tube realization, is used in the linear prediction component of the speech synthesizer, following a conversion from coded reflection coefficients back to the coefficients a_i. The error is quantized to 1 bit/sample (the sign bit) and packed into the bit stream in the modem transmit buffer. The bits are then shipped across the channel to the receiver, as shown in Fig. VII-2, the synthesizer unpacks and decodes the parameters, and reconstructs synthetic speech which, in the absence of channel errors, is identical to the s(n) previously computed by the analyzer.

D. DETAILS OF THE LINCOLN DVT (LDVT) IMPLEMENTATION

The APC implementation, unlike the LPC implementation, has essentially no real-time computation. In the real-time program, a new sample is received from the A/D converter and stored in the input buffer which is located in the LDVT's outboard memory. A new sample is fetched from the output buffer in this memory and sent to the D/A converter, and the modems (both transmit and receive) are checked and serviced if ready. In order to assure that no bits are dropped due to inadequate sampling of the modem, since the bit rate is higher than the sampling rate, the A/D converter is set to twice the sampling rate necessary, and on the odd numbered interrupts only the modem is serviced.

The problem of drift between the modem clock and the A/D clock is handled quite differently than in the LPC program. Adjustment due to slippage of the pointers in both the analyzer and the synthesizer is done by either skipping an entire frame or repeating an entire frame twice, but only during silence periods. Pointers remain in a danger region for a sufficiently long time that at least one silence frame essentially always occurs before the pointers would collide.

It was decided to incorporate an elaborate synchronization algorithm into the APC program which is capable of resynchronizing in approximately 5 frame times plus round-trip path delay. Both analyzers send in each frame a 2-bit synchronization code which is verified by each receiver. If a receiver finds 3 frames in which the 2-bit message was incorrect, it assumes that it has lost synchronization and responds by sending a 32-bit special code to the other LDVT. When this code is detected, by a matched filter routine which is always checking, a new frame is started at the end of the 32-bit message and the 32-bit special code is then sent to the first LDVT. Finally, these 32 bits are detected and the other LDVT resynchronizes.

Since there is no real-time processing in the APC implementation, the program structure is a straightforward sequence of subroutines corresponding to the block diagrams in Figs. VII-1 and VII-2. In Fig. VII-3, these subroutines are listed and their memory and time requirements are given. The most costly algorithm in terms of time is the AMDF pitch detection. If time were tight, one could easily cut the AMDF time in half by only allowing even values of pitch, at some slight degradation in quality.

As in the LPC implementation, program memory and data memory are essentially exhausted. Some memory could be gained, if necessary, by replacing the matched-filter synchronization algorithm with something simpler.

APC is characterized by a large number of speech buffers because the pitch filter necessitates a delay of M samples. Both the synthetic speech and the input speech are double buffered in the outboard memory. In each case, one buffer is a slave to the A/D-D/A while the other is used in the processing of the next frame. In addition, the analyzer's feedback loop requires a buffer of the previous M samples, but need not be double-buffered since the synthetic speech produced in the analyzer is not sent to the D/A converter. The current implementation requires 3 buffers of length N + PMAX (PMAX is the maximum allowable pitch period) and two of length N, and uses up 65 percent of the outboard memory.

Details of the algorithm are available in Volume II, Program Listings.

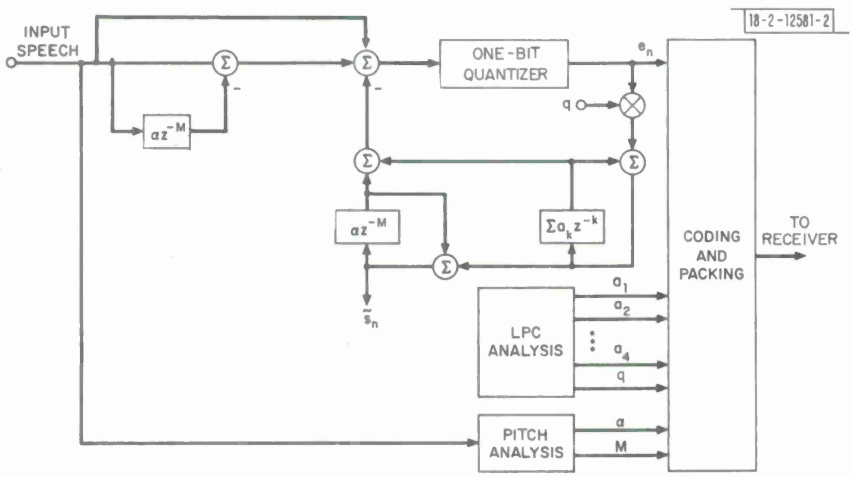


Fig. VII-1. The APC transmitter.

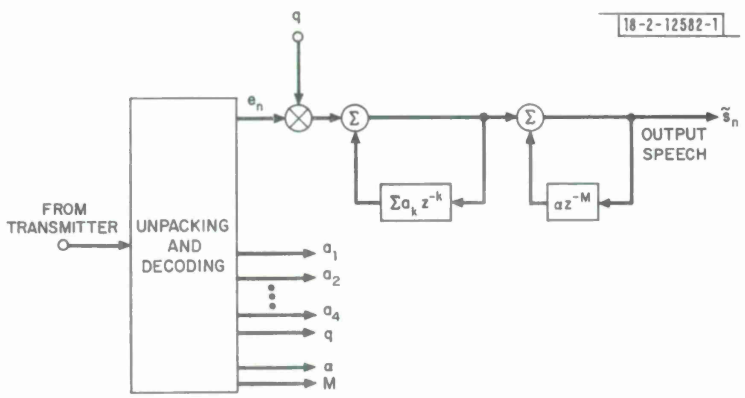


Fig. VII-2. The APC receiver.

18-2-12651

ALGORITHM	PROGRAM MEMORY	DATA MEMORY	OUTBOARD MEMORY	EXECUTION TIME (Msec)	
Buffer Handling	62	N + PMAX = 288	0	0.84	
Pitch Extraction	37	7	N + PMAX = 288	13.3	
Computation of a	56	5	–	0.27	
Correlation	80	14	–	0.84	
$d(n) = s(n) - as(n-M)$	11	0	–	0.10	
Levinson Recursion	134	20	–	0.035	N = 68 PMAX = 120
Coding	161	60	64	0.12	
Analysis Feedback Loop	67	10	N + PMAX = 288	0.82	
Decoding	87	60	64	0.05	
Synthesis	61	20	N + PMAX = 288	0.72	
A/D-D/A, Modem	75	9	2N = 366	0.77	
Matched Filter, Synchronization	154	8	–	0.55	
TOTAL	985 = 96%	501 = 99%	1358 = 65%	18.415 = 74%	

Fig. VII-3. APC breakdown.

VIII. THE LINEAR PREDICTIVE VOCODER

A. GENERAL DESCRIPTION OF THE ALGORITHM

LPC was first described by Atal and Hannauer in 1971.[1] Since then, many variations on this algorithm have appeared in the literature (see bibliography in Refs. 2 and 3). We have chosen to implement the Markel form of the LPC algorithms for reasons detailed in Ref. 4.

This algorithm is described in block-diagram form in Fig. VIII-1. Speech samples taken every 132 μsec are divided into 158 point groups corresponding to approximately 20 msec of data. These groups are multiplied by a Hamming window and then used to form P + 1 autocorrelation coefficients $R_o, \ldots, R_P$. The parameter P is the order of the filter used to model the vocal tract, and ranges from 10 to 12 in current LPC systems.

The autocorrelation coefficients are used as the constants in a set of linear equations that must be solved to obtain the parameters of the vocal-tract filter. These equations are solved by means of the Levinson recursion[5] which yields a set of P reflection coefficients $K_o, \ldots, K_{P-1}$ and a residual energy E. These reflection coefficients will be used at the receiver to implement the vocal-tract filter. The structure chosen for this filter is the acoustic tube filter described in detail in Ref. 2. The residual energy is used at the receiver to generate the amplitude of the excitation for the acoustic tube.

In addition to the processing described above, the raw speech samples are fed to a pitch and voicing detector which produces both a voiced-unvoiced decision and an estimate of pitch. The particular algorithm used for this purpose is the Gold-Rabiner pitch detector which is described in detail in Sec. IX.

The parameters produced as described above are next coded by means of a logarithmic-search table-look-up procedure and formed into a serial bit stream for transmission to the remote receiver. The receiver portion of the algorithm accepts such a serial bit stream from the remote transmitter and unpacks it to form the code-book addresses of the various parameters. These addresses are then decoded to obtain the actual values of the parameters which are then used to implement the acoustic tube filter and its excitation. The output of the filter is the final synthetic speech.

B. DETAILS OF THE LINCOLN DVT (LDVT) IMPLEMENTATION

The LDVT program for realizing the LPC algorithm consists of two major pieces: a real-time program which is interrupt-driven by the A/D converter and handles those computations that must be made every time a new speech sample is received, and a non-real-time program which handles those computations that must be made only when a complete frame of speech has been received.

The main task of the real-time program is to update the windowed correlator, the six elementary pitch detectors, and the synthesizer filter. The details of the pitch detector update are presented in Sec. IX. The synthesizer update consists of generating a sample of white noise whose amplitude is governed by the residual energy E if the frame is unvoiced, or the generation of either a zero or a pitch pulse of appropriate amplitude if the frame is voiced. The resultant excitation is then used to update the acoustic tube algorithm, thus producing a synthetic speech sample which is fed to the D/A converter.

In addition to the synthesizer update just described, the synthesizer parameters are interpolated in two distinct ways during a frame. The energy parameter used to drive the acoustic

tube is linearly interpolated every time a pitch pulse is generated. This results in smooth amplitude transitions from frame-to-frame and noticeably improves the quality of the synthetic speech. No energy interpolation takes place during unvoiced frames. In addition to this pitch-synchronous energy interpolation, the reflection coefficients that define the acoustic tube are linearly interpolated at regular intervals of about 5 msec. This avoids abrupt changes in the parameter values from frame-to-frame.

The correlation update is somewhat more complicated because of the requirement to produce complete 158-point correlations at a flexible rate. The need for this flexibility will be discussed later; the method used to achieve it was to start a new correlation every 158-S points rather than every 158 points. This means that more than one correlation must be updated at each interrupt but, as long as S is held less than 79, no more than two correlations must be updated at each interrupt. This provides a frame rate flexibility of 96 to 48 Hz, which is more than adequate for our needs.

The update of a single correlator is accomplished by first multiplying the incoming speech sample by its appropriate window value. The windowed speech sample is then pushed down on a stack of the previous P + 1 such samples. The k^{th} (k = 0, . . . , P) running correlation sum is then updated by adding to it the product of the most recent addition to the stack with the k^{th} entry of the stack. This addition is done with double-precision arithmetic; the full double-length stack product is added to the double-length running correlation sum. This process is facilitated by special LDVT double-precision instructions.

When the correlation routine determines that a 158-point double-precision correlation has been finished, it sets a flag that tells the non-real-time program to start its computation as soon as the real-time program has finished its current updates.

The basic tasks of the non-real-time program are the Levinson recursion, determination of pitch from the current state of the six elementary pitch detectors, and coding and framing. The Levinson recursion is straightforward and the final determination of pitch is described in Sec. IX. The Levinson recursion is done with single-precision arithmetic; however, the necessary correlation coefficients are presented to it in block-floating-point format. A special routine left-justifies the double-precision R_0 given it by the correlator and produces single-precision, block-floating-point correlation coefficients. The divisions required by the Levinson recursion are handled by an exact, but fairly slow (5 μsec), divide subroutine.

The coding of the parameters produced by the non-real-time analysis, except for pitch which is transmitted as is, is accomplished by a logarithmic-search table-look-up routine. The residual energy is logarithmically coded to 5 bits. The reflection coefficients are coded by means of truncated, log-area ratios in which each reflection coefficient is first clamped to an individually selected interval, transformed by the log-area-ratio function $\{\log [(1 - K)/(1 + K)]\}$, and finally truncated to the desired number of bits. The detailed bit assignments for the reflection coefficients are 6, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5 at 9800 and 3600 bps, and 6, 6, 4, 4, 3, 3, 3, 3, 2, 2 at 2400 bps. The detailed coding tables used to achieve these rates are included as part of the LPC program printout presented in Volume II.

After coding, the code-book addresses of the various parameters are packed into 16-bit words and delivered to the output buffer which is emptied by the parallel-to-serial converter and delivered to the transmit modem.

Since the transmit modem absorbs bits at an average rate determined by its internal clock, the analyzer portion of the LPC algorithm must adjust the average rate at which it produces bits

accordingly. The latter rate is governed by the number of code bits assigned each frame and the independent A/D converter clock. Equality between these two rates is achieved by dynamic adjustment of the frame rate. This is the reason for the requirement that the real-time correlator be able to produce new correlations at arbitrary intervals.

Frame-rate control is achieved by means of a "bang-bang" servo technique. The locations of the buffer pointers loading and unloading the output buffer are monitored once each frame. The difference between these two pointers determines whether the overlap parameter S controlling the frame rate should be left as is or set to produce a higher or lower frame rate. This strategy guarantees that, on the average, the number of bits/second produced by the analysis program matches the number of bits/second being taken by the modem.

A similar tactic is employed by the real-time synthesis portion of the program which must insure that the rate at which it uses up bits matches, on the average, the rate at which the receiver modem is supplying them. Here, control is exerted by monitoring the input-buffer loading and unloading pointers and using their difference to determine for how many samples the current synthesis should continue before a new set of synthesis parameters are derived from the input buffer.

The final details of the LPC algorithm are summarized in Fig. VIII-2 which depicts the running times and memory requirements of the various components of the algorithm. The salient points to be made here are that, with regards to running time, the machine is at least twice as fast as required for LPC, and just adequate as far as program and data memory requirements are concerned if overlay techniques are to be avoided. The use of overlays, however, enables the LDVT to execute considerably more demanding algorithms.

REFERENCES

1. B. S. Atal and S. L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave," J. Acoust. Soc. Am. 50, 637 (1971).
2. J. D. Markel et al., "Linear Prediction of Speech – Theory and Practice," SCRL Monograph No. 10, Speech Communications Research Laboratory, Inc., Santa Barbara, California (September 1973).
3. J. D. Makhoul and J. J. Wolf, "Linear Prediction and the Spectral Analysis of Speech," BBN Report No. 2304, Bolt, Beranek and Newman, Inc., Cambridge, Massachusetts (August 1972).
4. J. D. Markel and A. H. Gray, Jr., "A Linear Prediction Vocoder Simulation Based upon the Autocorrelation Method," IEEE Trans. Acoustics, Speech, and Signal Processing ASSP-22, 124 (1974).
5. N. Wiener, Extrapolation, Interpolation and Smoothing of Stationary Time Series (The Technology Press and J. Wiley and Sons, New York, 1957), Appendix B.

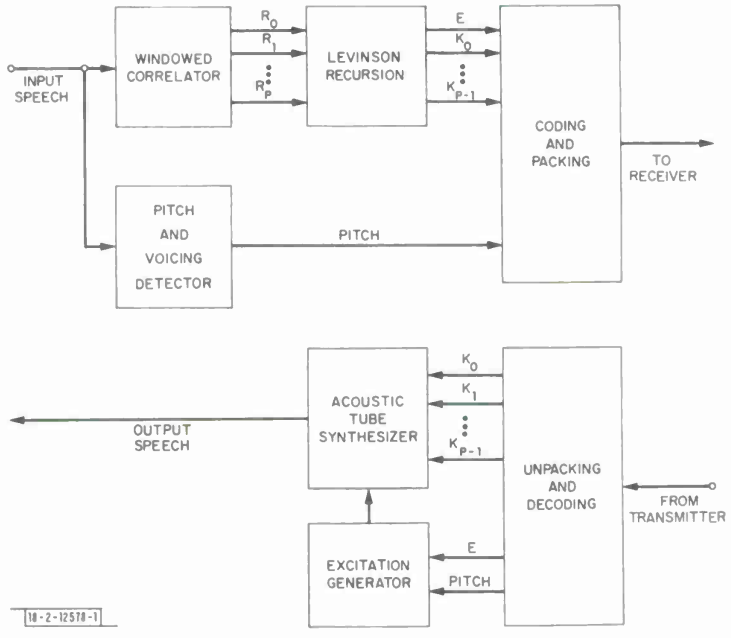


Fig. VIII-1. The LPC vocoder.

18-2-12852

ALGORITHM	PROGRAM MEMORY	DATA MEMORY	OUTBOARD MEMORY	EXECUTION TIME (μsec)	PER SAMPLE OR PER FRAME
Double-Windowed Correlator	109	88	79	22	PS
Real-Time Pitch	152	95	–	10 to 23	PS
Acoustic Tube Synthesis	72	67	–	8	PS
Levinson Recursion	142	42	–	220	PF
Pitch Determination	141	50	–	545	PF
Coding/ Decoding	59	56	467	95	PF
Framing/ Deframing	110	32	–	60	PF

Fig. VIII-2. LPC breakdown.

IX. PITCH-DETECTION STUDIES

The Gold-Rabiner pitch detector* has been incorporated in a real-time linear predictive system for use in both the FDP and the DVT. The system in the FDP operates on a complete frame of data while buffering incoming samples. In the DVT, data are processed sample-by-sample in the foreground program, while frame-by-frame processing takes place in the background. The adaptability of the pitch detector to these two types of real-time environments will be described.

The Gold-Rabiner pitch detector consists primarily of a digital low-pass filter, a peak processor, six individual pitch-period estimators, and a final pitch-period computation (Fig. IX-1). Whenever a peak is detected, three of the six period estimators become activated, depending on whether the peak is positive or negative. Three distinct measurements are made upon the occurrence of a peak (Fig. IX-2) – either M1, M2, and M3 for a positive peak, or M4, M5, and M6 for a negative peak. Measurements M1 and M4 are actual peak values. Measurements M2 and M5 are peak-to-valley measurements, and M3 and M6 are peak-to-peak measurements. Each of the three measurements is compared with its particular threshold which is a function of the previous successful measurement. When a measurement succeeds, the pitch period thus detected (i.e., the number of samples since the last successful measurement) is recorded in that period estimator's channel information block, and the current measurement becomes the new threshold. A new period average (P AV) is determined from the previous P AV and the new period. Based on P AV, two parameters are set in that channel block: (a) a blanking count (some number of samples) during which no measurements will be honored, and (b) a decay factor which will act upon the threshold once the blanking count has been exhausted. At any point in time a channel information block has recorded in it the latest three pitch periods detected by its period estimator.

When a frame of speech data has been processed, a 36-element matrix of pitch periods is formed using the 18 periods from the six channel blocks plus the double and triple summations of the contiguous pitch periods. The six most recent periods, one from each channel, become the six pitch-period candidates which are compared with the 36-period matrix, using a scoring algorithm described later. As a result of this score, the frame of speech is declared voiced or unvoiced and, if voiced, a pitch period is chosen.

The filter currently used in the pitch detector is a finite impulse response low-pass filter (Fig. IX-3) with a frequency response shown in Fig. IX-4. A filtered sample is the result of an average over the most recent ten samples using the following algorithm. The ten samples are summed five at a time, from which six averages are computed. These six averages are summed four at a time, from which three averages are computed. The average of these three averages becomes the filtered output sample. When this filter was substituted for an 8-pole recursive filter which had previously been used in the pitch detector, it was observed that the time spent in the peak processor was reduced to about one-third. No degradation in the accuracy of the pitch was noticed.

The FDP is a parallel processor with 4 arithmetic elements, 16 index registers, 512 program memory locations, 4096 data locations, 164,000 additional memory locations accessible via the I/O system, a cycle time of 150 nsec, and a program overlay facility. The interaction

* B. Gold and L. R. Rabiner, "Parallel Processing Techniques for Estimating Pitch Periods of Speech in the Time Domain," J. Acoust. Soc. Am. 46, 442 (1969).

with peripheral devices is restricted to direct interrogation without interrupts. The real-time linear predictive system implemented on the FDP uses double-buffering for the input and output speech. Processing is done on a frame-by-frame basis with a sprinkling of interrogations to the A/D converter throughout the program. If the difference between the maximum and minimum filtered samples within a frame is below the energy threshold, that frame is declared unvoiced and the peak search is bypassed. The real-time pitch detector in the FDP occupies 335 program locations, the latter making it necessary to overlay portions of the linear predictive program. With a frame of 110 samples and a sampling interval of 150 μsec, the average processing time of the pitch detector is 4.65 msec per frame or 42 μsec per sample. The final scoring of the pitch-period candidates accounts for about 2 msec per frame, the filter and energy check for 1.5 msec; and from actual timing measurements, the peak processor was observed to average about 1 msec per frame. No pitch-period editing is done in the real-time pitch detector.

The decision was made to merge the linear predictor with another method of pitch detection involving an absolute magnitude difference function (AMDF). The AMDF is of the form:

$$\sum_{n=1}^{N} |S_n - S_{n+p}| = f(p)$$

where p varies over a range of possible pitch periods, and n is incremented in steps of four. The estimated period is the p for which the function is a minimum. This particular pitch detector consists of a test for silence via zero-crossings, a filter (the same filter used with the Gold-Rabiner pitch detector), computation of the energy within the frame, normalization of the data, the AMDF, and an edit of the period chosen two frames previously. To incorporate this AMDF pitch detector, it was necessary to increase the size of the raw speech buffer and to delay the linear predictor parameters one more frame. Accuracy of pitch is somewhat sacrificed by the quantized set of periods used in the AMDF, but this may be corrected for by the pitch-period smoothing in the final editing. The AMDF pitch detector as programmed for the FDP has a tendency to cause premature unvoicing, possibly as a result of approximating some of the parameters indicated in the original algorithm (the FDP does not have a floating-point hardware package or a divide instruction). The estimated running time of the AMDF pitch detector is comparable to that of the Gold-Rabiner pitch detector, but a large buffer of speech must reside in memory in order to perform the AMDF. The AMDF pitch detector occupies 492 program locations and uses 374 data locations. The Gold-Rabiner pitch detector occupies 335 program locations and uses 189 data locations.

The structure of the Gold-Rabiner pitch detector was easily separated into real-time and non-real-time processing for implementation in the DVT. The DVT has one arithmetic element, one index register, 1024 program memory locations, 512 data locations, 2048 additional memory locations accessible via the I/O system, a cycle time of 55 nsec, and a program overlay facility. Peripheral devices may be interrogated directly or be set to interrupt; however, only one device may be active at any time. Filtering and peak processing are done sample-by-sample in the interrupt routine, and the final scoring and pitch-period decision are activated by the linear predictive program when a frame of data has been processed (Fig. IX-5). The maximum and minimum values of the filtered data are updated by the real-time program, and these values are checked and reset by the non-real-time processor to determine the energy level of the frame. Sampling at 132-μsec intervals, the estimated running time of the real-time processor ranges from about 10 to 23 μsec, depending on whether or not a peak is found.

With a frame length of 150 samples, the non-real-time portion of the pitch detector averages 3.63 μsec per sample. The entire pitch processor occupies 295 program locations and uses 145 data locations, 49 of which may be shared. As in the FDP, no pitch editing is done.

The Gold-Rabiner pitch detector can be incorporated into two kinds of real-time speech processing as has been shown. Its adaptability to a small computer with a limited internal storage capacity, its performance reliability even without pitch-period editing, and the ease with which it can be implemented have made it a valuable aid in the experimentation in speech compression done at Lincoln Laboratory.

DETAILED DESCRIPTION OF FRAME-BY-FRAME PITCH DETECTOR

A. Filter N-sample frame.

B. Determine the maximum and minimum values of the filtered speech signal within the frame, and check the difference against an energy threshold which should be set to accommodate the audio system being used. If the energy is low (i.e., the frame is silence), set the voiced/unvoiced indicator to unvoiced, and set the periods and number of samples since the last successful peak to the initial values (listed under INITIAL CHANNEL INFORMATION) in each of the six channel blocks. If the energy is sufficiently high, perform the peak search.

C. Search sample-by-sample for peaks. When a change in slope occurs, take the previous sample as the peak value. If it is a negative peak, complement the value (this result may be negative if the value of the peak is positive). Store the peak value as the current positive or negative peak and take the measurements described below. After each sample, decrement the blanking count if >0, increment the number of samples since the last success, and update the current measurement threshold (threshold = old threshold × decay factor) if the blanking count has reached zero. Do this for each of the six channel information blocks, and return to peak search.

D. Take measurements M1, M2, M3 (positive peak) or M4, M5, M6 (negative peak) and store in respective channel blocks:

M1, M4: peak value = current positive or negative peak

M2, M5: peak ← → valley = current positive peak + current negative peak

M3, M6: peak-to-peak = current peak value − previous peak value

(See Fig. IX-2)

Check each of the three measurements as follows. If the blanking count is not equal to zero or the measurement is less than the threshold, call the measurement a failure and proceed to the next measurement. If the measurement is a success, store it as the new threshold, slide periods P_A and P_B to periods P_B and P_C, and store the number of samples since the last success as period P_A. If the previous frame was unvoiced, do not change P average (P AV). If the previous frame was voiced, compute new P AV = (old P AV + P_A)/2 and confine to a range which is a function of the sampling interval as discussed later. Compute blanking count = 0.4 (P AV), store the appropriate decay factor, and set the number of samples since the last success to zero. Proceed to the next measurement.

E. At the end of the frame, form a table of 36 pitch periods by storing P_A, P_B, P_C, $P_A + P_B$, $P_B + P_C$, and $P_A + P_B + P_C$ from each of the channel information blocks. The six

pitch-period candidates are the most recent periods, P_A, from the six channels. The pitch period of each candidate being tested determines the window of tolerance. This window is a function of the sampling rate as shown later. A window has four "panes" with associated biases. Each pitch-period candidate, P_K, is compared with all 36 values four times as follows:

(1) Clear pitch-period score (PSCORE) for this candidate.

(2) Clear score counter (SCORE). If $|P_K - P_N| \leq PANE_K$, increment SCORE, N = 1, . . . , 36.

(3) Add bias for this window pane to SCORE.

(4) Compute NEW SCORE = SCORE-THRESHOLD (The current algorithm uses a threshold of 13.)
Compare magnitudes of NEW SCORE and PSCORE. If |NEW SCORE| > |PSCORE|, replace PSCORE with NEW SCORE.
(Note that NEW SCORE may be negative.)

(5) Repeat (2) through (4) using remaining panes.

(6) Save PSCORE for this candidate.

(7) Repeat (1) through (6) for remaining pitch-period candidates.

F. Pick the winning pitch period from the six candidates by choosing the highest score. If the winning score is negative or if the winning pitch is $\geq P_{MAX}$, set the voiced/unvoiced indicator to unvoiced. (P_{MAX} = 176, 128, 117 for sampling intervals of 100, 130, and 150 μsec, respectively.) If the winning pitch period is accepted, set the voiced/unvoiced indicator to voiced.

EXAMPLE OF SCORING ALGORITHM USING WINDOWS FOR 130-μsec CHANNEL INFORMATION AT END OF FRAME

	1	2	3	4	5	6
P_A	70	71	35	71	74	36
P_B	71	70	35	70	70	33
P_C	69	70	33	71	69	71

36-PERIOD MATRIX

P_A	70	71	35	71	74	36
P_B	71	70	35	70	70	33
P_C	69	70	33	71	69	71
$P_A + P_B$	141	141	70	141	144	69
$P_B + P_C$	140	140	68	141	139	104
$P_A + P_B + P_C$	210	211	103	212	213	140

SCORE RESULTS

P_A	70	71	35	71	74	36
SCORE	10	11	−7	11	9	−7

CHANNEL INFORMATION KEPT FOR EACH OF SIX PITCH DETECTORS

(1) P_A
(2) P_B
(3) P_C
} last 3 periods detected

(4) No. samples since last successful peak
(5) Current measurement
(6) Previous measurement (needed only for M1 and M4)
(7) P AV
(8) Blanking count
(9) Current measurement threshold
(10) Decay factor

INITIAL CHANNEL INFORMATION

(1) 200
(2) 250
(3) 300
(4) 350
} arbitrary values for nonperiodicity

(5) 0
(6) 0
(7) Lower limit of P AV
(8) 0
(9) 0
(10) Decay factor for P AV

DECAY FACTORS AS A FUNCTION OF SAMPLING INTERVAL

(Decay Factor = $e^{-0.695/P}$)

100 μsec: $40 \leqslant P\,AV \leqslant 100$, limits of P AV

IF P AV <	P =
48	44
56	52
64	60
72	68
80	76
88	84
96	92
104	100

130 μsec: $30 \leqslant P\,AV \leqslant 77$

IF P AV <	P =
36	33
42	39
48	45
54	51
60	57
66	63
72	69
78	75

150 μsec: $26 \leqslant P\,AV \leqslant 66$

IF P AV <	P =
32	29
37	34
42	39
47	44
52	49
57	54
62	59
67	64

WINDOWS OF TOLERANCE AS A FUNCTION OF SAMPLING INTERVAL

100 μsec

PANES

PITCH-PERIOD RANGES	1	2	3	4	
16-31	1	2	3	4	WINDOW 1
32-63	2	4	6	8	WINDOW 2
64-127	4	8	12	16	WINDOW 3
128-255	8	16	24	32	WINDOW 4

130 μsec

PANES

PITCH-PERIOD RANGES	1	2	3	4	
12-24	1	2	3	4	WINDOW 1
25-48	2	4	6	8	WINDOW 2
49-98	3	6	9	12	WINDOW 3
99-196	6	12	18	24	WINDOW 4

150 μsec

PANES

PITCH-PERIOD RANGES	1	2	3	4	
10-20	1	2	3	4	WINDOW 1
21-42	2	4	6	8	WINDOW 2
43-84	3	6	9	12	WINDOW 3
85-170	5	10	15	20	WINDOW 4

PANE BIAS =

8	6	3	1

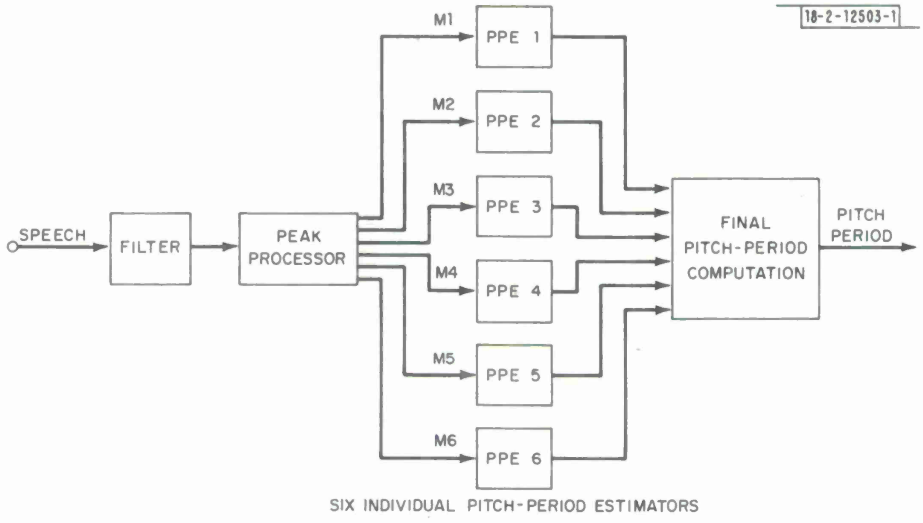


Fig. IX-1. Block diagram of pitch-period estimation algorithm.

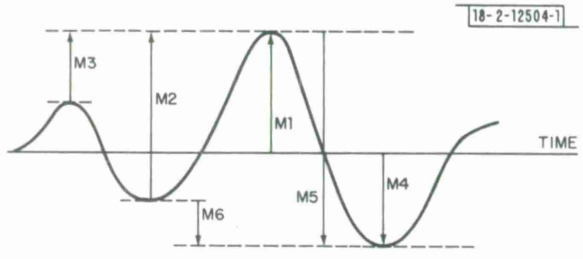


Fig. IX-2. Basic measurements made on filtered speech.

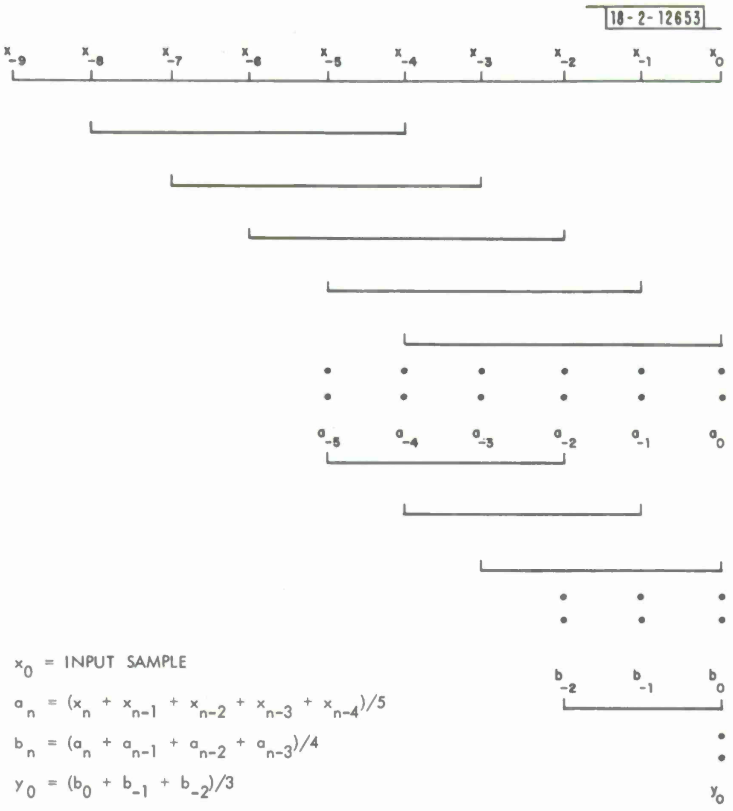


Fig. IX-3. Low-pass filter used in pitch detector.

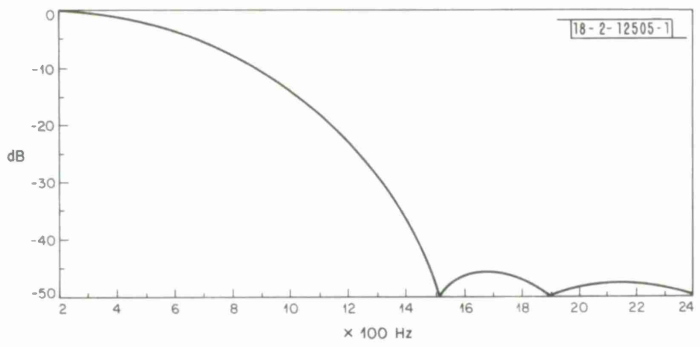


Fig. IX-4. Frequency response of filter at 132-μsec sampling interval.

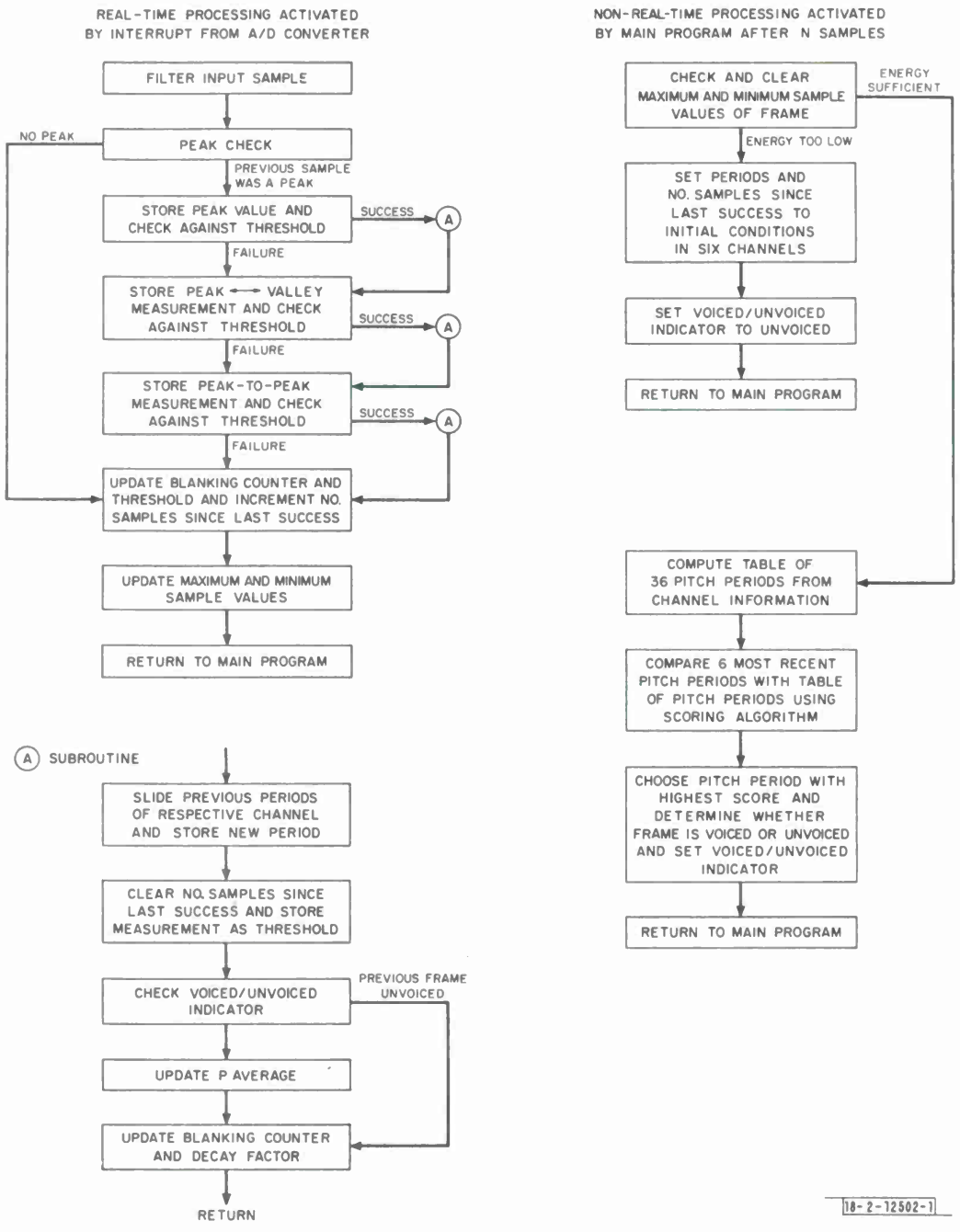


Fig. IX-5. Flow diagram of Gold-Rabiner pitch detector.

X. CONTINUING WORK AND CONCLUSIONS

This report describes work performed during FY 75 by Lincoln Laboratory's Group 24 under the DCA Speech Evaluation contract. It is clear that the program had a two-fold thrust: a software effort which investigated several speech encoding algorithms, and a hardware effort which produced two general-purpose, high-speed, signal-processing computers. In the software area, real-time simulation allowed for refinement of algorithms at 16, 9.6, 4.8, 3.6, and 2.4 kbps at first on the FDP and then running in the DVT. In a sense, the FY 75 effort is crystallized in the DVT as a flexible speech coding "black box" capable of running many algorithms by changing ROM storage. The two efforts of software algorithm research and hardware development thus are combined. Many areas of useful investigation remain in algorithm development, in speech network design (i.e., conferencing), and in special-purpose speech-digitizing hardware.

Since all this research and development work has been and will continue to be guided by the Secure Speech Consortium, let us list Consortium goals for FY 76 and then list the Lincoln Laboratory effort for FY 76.

In FY 76, the Consortium will perform four major tasks:

(a) Continue test and evaluation of digital voice terminals with emphasis on 16 kbps.

(b) Move toward advanced development models of narrow-band (2.4-, 3.6-, and 4.8-kbps) voice encoders.

(c) Study interactions between 16- and 2.4-kbps digitizers so that a tandem arrangement is acceptable in quality and intelligibility.

(d) Study conferencing techniques.

In FY 76, Lincoln Laboratory will perform the following tasks:

(a) Tandeming and conferencing experiments using high-, medium-, and low-rate digitizers.

(b) Telephone channel simulation for controlled testing of phone-line distortions and their effect on speech digitizers.

(c) Investigations of speech digitizer talker sensitivity, and techniques for reducing this effect.

(d) Study medium rate coders with a view toward improved quality and/or lower implementation costs.

(e) Design and develop a low-cost LPC microprocessor terminal for use at 4.8, 3.6, and 2.4 kbps. Such a terminal must be suitable for large-scale defense communications systems deployment.

(f) Investigate effects and cures for carbon button microphone inputs to speech digitizers.

(g) Investigate effects and cures for input environmental noise vulnerability in speech digitizers.

All these tasks are directly or indirectly related to Consortium and/or DCA interests and should lead to next-generation voice digitizers which will be robust with respect to input speech distortions, channel effects, and tandeming. At the same time, new microprocessor realizations for these devices will allow for low-cost implementation suitable for large-scale network use.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-75-297	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Speech Evaluation		5. TYPE OF REPORT & PERIOD COVERED Annual Report, FY 1975 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR*(s)* Gold, Bernard		8. CONTRACT OR GRANT NUMBER*(s)* F19628-73-C-0002
9. PERFORMING ORGANIZATION NAME AND ADDRESS Lincoln Laboratory, M.I.T. P.O. Box 73 Lexington, MA 02173		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Element No. 30100
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Communications Agency 8th Street & So. Courthouse Road Arlington, VA 22204		12. REPORT DATE 30 June 1975 13. NUMBER OF PAGES 68
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* Electronic Systems Division Hanscom AFB Bedford, MA 01731		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

speech evaluation	speech-processing systems	voice-excited systems
speech compression	Lincoln Digital Voice Terminal	pitch detection
vocoder systems	microprocessor chip sets	hybrid packaging

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Volume I reports the work performed during FY 75 on the DCA-Speech Evaluation Contract. Two broad categories of work are described: algorithmic studies and hardware design.

Several algorithms for digitizing and reducing the data rate of speech signals are described. These algorithms include an adaptive residual coder (ARC) designed to produce data at 16 and 9.6 kbps, an adaptive predictive coder (APC) at 8 kbps, a voice-excited linear predictor (VELP) at 8 kbps, and a straight linear predictive coded (LPC) vocoder at 2.4, 3.6, and 4.8 kbps. In addition, some work on pitch or excitation extraction is described. All these studies are evaluated on a real-time facility which is described.

In the hardware design area, the digital voice terminal is described in detail, as well as some follow-on next-generation LSI studies.

Volume II will contain a DVT manual, program listings, and a cross assembler.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*